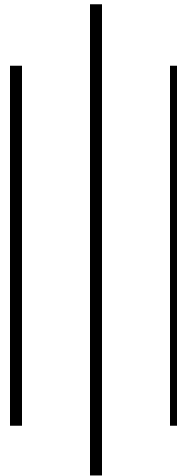




Tribhuvan University
Institute of Engineering
Pulchowk Campus



A FINAL YEAR PROJECT REPORT ON
“CDMA BASED PERSONAL COMMUNICATION SYSTEM”

SUBMITTED BY:

Rikesh Shakya (061/BEX/434)
Sabin Maharjan (061/ BEX /436)
Sudat Tuladhar (061/ BEX /441)
Sujan Raj Shrestha (061/ BEX /443)

SUBMITTED TO:

Department of Electronics and
Computer Engineering,
Pulchowk Campus, IOE,
Lalitpur, Nepal.

Date: March 31, 2009



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

Ananda Niketan, Pulchowk, Lalitpur, P.O.Box 1175, Kathmandu, Nepal.

Tel : 5534070, 5521260 extn. 311, Fax : 977-1-5553946, E-mail : ece@ioe.edu.np

Our Ref :



Date: March 31, 2009

DEPARTMENT ACCEPTANCE

The project entitled "**CDMA based Personal Communication System**" submitted by Mr. Rikesh Shakya, Mr. Sabin Maharjan, Mr. Sudat Tuladhar and Mr. Sujan Raj Shrestha in partial fulfillment of the requirements for the degree of Bachelors of Engineering in Electronics and Communication has been accepted as a bona fide record of the work carried out in this department.

S. Joshi

Head of Department

(Prof. Dr. Shashidhar Ram Joshi)

Department of Electronics and Computer Engineering
Institute of Engineering, Pulchowk Campus.



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

Ananda Niketan, Pulchowk, Lalitpur, P.O.Box 1175, Kathmandu, Nepal.

Tel : 5534070, 5521260 extn. 311, Fax : 977-1-5553946, E-mail : ece@ioe.edu.np

Our Ref :



Date: March 31, 2009

The project entitled "**CDMA based Personal Communication System**" submitted by Mr. Rikesh Shakya, Mr. Sabin Maharjan, Mr. Sudat Tuladhar and Mr. Sujan Raj Shrestha in partial fulfillment of the requirements for the degree of Bachelors of Engineering in Electronics and Communication has been accepted as a bona fide record of the work carried out in this department.

Supervisor,
Prof. Dr. Dinesh Kumar Sharma
Institute of Engineering

Internal Examiner,
Mr. Sudip Rimal
Lecturer
Institute of Engineering

Supervisor,
Mr. Pradeep Poudyal
Lecturer
Institute of Engineering

External Examiner,
Mr. Lochan Lal Amatya,
Manager, IT Directorate,
Nepal Telecom.

Head of Department
Prof. Dr. Shashidhar Ram Joshi
Department of Electronics and Computer
Engineering

Project Co-ordinator,
Dr. Surendra Shrestha,
Department of Electronics and Computer
Engineering

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge for all the help and co-operation we received during this project. We would like to convey our gratitude to **Department of Electronics and Computer Engineering** for providing us the necessary infrastructure for the continuity of this project. We are thankful to Final Project Co-coordinator **Dr. Surendra Shrestha, Project Coordinator** for his kind support and co-operation.

The project wouldn't have been so progressive without the guidance of **Prof. Dr. Dinesh Kumar Sharma** and **Mr. Pradeep Poudyal, Project Supervisors**. We are immensely thankful to both of our supervisors for all their help and guidance and each time they helped us out during problematic situations. We should never forget to acknowledge **Prof. Dr. Shashidhar Ram Joshi, Head of Department, Department of Electronics and Computer Engineering**, without whom the accomplishment of even the simplest task would be impossible. We are indebted to our **Mr. Lochan Lal Amatya, Manager, IT Directorate, Nepal Telecom, External examiner**, whose valuable suggestions and guidance helped us to explore and express the objectives more clearly. We would like to extend our sincere gratitude to our respected teacher and internal examiner **Mr. Sudip Rimal, Internal Examiner** for his support and suggestions.

We are also thankful to **Er. Binit Sharma** and **Er. Narendra Maharjan, Nepal Telecom** for their co-operation and kind support during our internship on CDMA.

Last but not the least, we would like to thank all our friends who have directly or indirectly contributed in this project, without their help, inspiration and support, our project would not be successful.

ABSTRACT

CDMA is a multiplexing technique in which all of the users use whole of the available radio channels for whole of the time. In CDMA, each user is assigned a unique code which is used for the separation of a particular user's data from others. It has many attractive features like high data rates, low power consumption, large coverage, high privacy, hard to wiretapping, decreased call-drop rate due to soft handoff, dynamic accommodation of users (soft capacity) etc. It does have a competitive advantage over other contemporary wireless technologies.

Since CDMA is optimized for higher data rates, smoother transitions can be made to the 3G era of mobile communications. The project is the first step in understanding the underlying principles behind CDMA and adopting it for future coursework and study.

The project titled “**CDMA Based Personal Communication System**” is a two user based communication system. The input data for users is fed through a PC. A Graphical user interface (GUI) is built using a MATLAB tool. Two user's data pass to the hardware (FPGA) via a serial interface and a parallel interface. In FPGA transmitter, the user data (size limited) is multiplied with respective orthogonal codes uniquely defined to each of the users; the process is called spreading. The results are then added and transmitted via a wired channel. On the receiver side, same orthogonal codes recover the original user data by de-spreading the received signal with the same orthogonal codes. The received data is displayed in a seven-segment.

Further, simulations of processes like spreading, de-spreading, channel encoding etc are carried out in MATLAB. The respective sliders form input for each user ranging from 0 to 9. Built-in functions are used to digitize the analog signal. Pertinent waveforms help build insight on processes as such.

Table of Contents

Chapter 1: Overview

1.1 Objectives:	1
1.2 Desired System:	1
1.3 Theory	1
1.3.1 CDMA Technology	2
1.3.2 Multiple Access	2
FDMA	3
TDMA	3
CDMA	3
1.3.3 Spread Spectrum Types	4
FHSS	4
DSSS	4
1.3.4 Spreading Codes	4
PN Sequence	4
Orthogonal Codes	5
1.3.5 Auto-Correlation and Cross-Correlation	5
1.3.6 CDMA operation from speech input to speech output	6
Source Coding	6
Channel Coding	6
Convolutional Coding	7
Viterbi Algorithm	7
Interleaving	9
Scrambling	9
Spreading	9
De-spreading	10
Spreading and de-spreading principle for two users	10
Modulation and Demodulation	11
1.3.7 Computer Interface	11
Parallel Interface	11
Serial Interface	12
1.3.8 Asynchronous Serial Communication	14
UART	14
UART Receiving Subsystem	15
MAX232	16
1.3.9 LCD Interfacing	16
1.3.10 Modern Digital Design using HDL	18
Digital Design Methodology using HDL	18
Features of modern HDL language	19
Use of an HDL program	19
Advantages of HDLs	20
Description of VHDL	20

1.3.11 Spartan II FPGA (XC2s50tq144-5)	22
XSA board Organisation	22
FPGA Programming	23
Chapter 2: Implementation Tools	
2.1 List of Tools and their brief utility	25
2.1.1 XC2s50 Spartan-II FPGA	25
2.1.2 HDL Language: VHDL	25
2.1.3 HDL Simulator: ModelSim XE 6.1c	25
2.1.4 Synthesis Tool for FPGA: Xilinx ISE Project Navigator	26
2.1.5 Simulation Software: MATLAB	26
Chapter 3: Hardware Design and Simulation	
3.1 Design Overview	27
3.2 Design of Communication Block-Sets targeted to FPGA	27
3.2.1 Transmitter	27
Orthogonal Code Generator	28
UART	28
Parallel to Serial Converter	28
Spreading Block	29
3.2.2 Receiver	31
Serial to Parallel Converter	31
Despreading Block	31
LCD Interfacing module	32
7-seg decoder module	32
3.2.3 Convolutional Encoder	35
3.3 Interfacing Computer Through Parallel and Serial Ports	36
3.4 Interfacing 7 Segment Display with Line Driver 74LS245	37
3.5 Final Circuit Diagram	38
Chapter 4: Software Simulation (MATLAB)	
4.1 Analog Simulation	40
4.2 Digital Simulation	41
Chapter 5: Cost Estimation	43
Chapter 6: Epilogue	44
5.1 Limitation	44
5.2 Future Enhancement	44
5.3 Difficulties faced in the project	44
5.4 Conclusion	45
System Operation Snapshots	
Appendix	
References	
Credits and Contact List	

List of Figures	Page No.
Figure 1.1: Multiple Access Technique.....	3
Figure 1.2: Four stage feedback register.....	5
Figure 1.3: CDMA Communication Model.....	6
Figure 1.4: Convolutional Encoder.....	7
Figure 1.5: (a) State Diagram and (b) trellis diagram of Convolutional Encoder.....	7
Figure 1.6 An encoding example of figure.....	8
Figure 1.7: Decoded Output of Trellis.....	8
Figure 1.8: Block Diagram of Viterbi Decoder.....	8
Figure 1.9: Spreading Process.....	10
Figure 1.10: Bit format used for sending asynchronous serial data.....	13
Figure 1.11: Transmission of a byte of serial data.....	15
Figure 1.12: UART Receiving Subsystem.....	16
Figure 1.13: MAX232 Pin Layout.....	16
Figure 1.14: VHDL Hardware Model.....	21
Figure 1.15: External connections to the XSA Board.....	22
Figure 1.16: FPGA Programming Process.....	24
Figure 3.1: Overall Schematic of the Transmitter.....	30
Figure 3.2: Receiver top level module.....	32
Figure 3.3: Receiver State Diagram.....	33
Figure 3.4: Simulation Results of Transmitting FPGA.....	34
Figure 3.5: Simulation Results of Receiving FPGA.....	34
Figure 3.6: RTL schematic of convolutional encoder.....	35
Figure 3.7: Simulation Results of Convolutional Encoder.....	35
Figure 3.8: Interfacing FPGA with PC's Parallel and Serial ports.....	36
Figure 3.9: Interfacing 7 Segment Display with Line Driver 74LS245.....	37
Figure 3.10: Final Phase Circuit Diagram.....	38

List of Figures	Page No.
Figure 4.1: Analog Simulation.....	40
Figure 4.2: Digital Simulation (with channel coding)	41
Figure 4.3: Digital Simulation (without channel coding).....	41

List of Tables	Page No.
Table 1.1: Pin Assignments of D-Type 25 pin Parallel Port.....	12
Table 1.2: Pin Connection of Serial Port.....	14
Table 1.3: Pin Description of 16*2 line LCD.....	17
Table 3.1: Interfacing PC and FPGA.....	36
Table 5.1: Cost Estimate of the project.....	43

List of Abbreviations

CDMA	Code Division Multiple Access
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
PC	Personal Computer
PN	Pseudo Noise
HDL	Hardware Description Language
VHDL	Very High Speed Integrated Circuit (VHSIC) Hardware Description Language
UART	Universal Asynchronous Receiver Transmitter
LCD	Liquid Crystal Display
FDMA	Frequency Division Multiple Access
TDMA	Time Division Multiple Access
FHSS	Frequency Hopped Spread Spectrum
DSSS	Direct Sequence Spread Spectrum
DS-CDMA	Direct Sequence Code Division Multiple Access
ASCII	American Standard Code for Information Exchange
GSM	Global System for Mobile
IS	Interim Standard
AM	Amplitude Modulation
FM	Frequency Modulation
SS	Spread Spectrum
PCM	Pulse Code Modulation
ESN	Electronic Serial Number
QPSK	Quadrature Phase Shift Keying
LSB	Least Significant Bit
I/O	Input Output
RTL	Register Transfer Logic
IEEE	Institute of Electrical and Electronics Engineers

CPLD	Complex Programmable Logic Device
DIP	Dual Inline Package
VGA	Video Graphics Array
CLB	Configurable Logic Block
LUT	Look up Table
ASIC	Application Specific Integrated Circuit
CAD	Computer Aided Design
GND	Ground
TTL	Transistor Transistor Logic
RF	Radio Frequency
DSP	Digital Signal Processing
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
LNA	Low Noise Amplifier
PA	Power Amplifier

Chapter 1: Overview

1.1 Objectives:

The objective of the project entitled “CDMA Based Personal Communication System” is to create a prototype model of a Code Division Multiple Access System for uni-directional data transfer between two pair of users. The following are main objectives of the project.

- To understand the CDMA data transmission and communication.
- To research on the principles behind the CDMA communication.
- To simulate the DS-CDMA using MATLAB tools.
- To implement the CDMA data communication in FPGA.
- To implement channel coding technique for reliable data transfer.
- To get acquaintance with VHDL, a powerful HDL language.

1.2 Desired System

Two user data are spread using respective orthogonal codes, combined and transmitted in a common channel which is received and de-spreaded using the same orthogonal codes to gain the original user data.

Two user data are supplied through PC. The graphical interface is built in a MATLAB tool. Two input data are given by changing the position of the slider in the MATLAB GUI. This not only forms the input for the transmitter realized in the Spartan II FPGA but also for the simulation in the MATLAB.

User data 1 is fed to the FPGA via the serial interface while the user data 2 is fed via the parallel interface. Spartan II FPGA does not have a serial port or hardware UART to receive the serial data. Hence a UART is realized by VHDL programming and the serial data with the standard format of 1 start bit and 1 stop bit with 8 data bits at the speed of 19,200 kbps is received in FPGA. Other user's data is provided through the parallel port.

Each user data is then spread using the respective orthogonal codes. ASCII representation of the data is used. Each data bit is first packed in a serial queue to be spread each by eight bit spread codes .Same is done for the next user data .Now two user data is added to be transmitted into a common wired channel. The orthogonal codes are pre-defined for each user in the VHDL programming.

In the receiver which is also realized in the Spartan II FPGA, same orthogonal codes are used to de-spread the user data to gain the original user data .The user data decoded as such is displayed in the seven segment display and LCD which is interfaced with the FPGA.

1.3 Theory

1.3.1 CDMA technology

Code division multiple access (CDMA) is a channel access method utilized by various radio communication technologies. The CDMA users have advantages over GSM and IS-136 TDMA that soft hand-off occurs. It consumes less battery power supporting upto 4 hrs of talk time. Other areas where CDMA are dominant over other technologies are voice quality, system reliability, and high data rate.

IS-95 standard is one very popular CDMA standard that is quite flexible, enabling service providers to allocate data in increments of 8 Kbps within 1.25 MHz. CDMA 2000 is another improved standard that provides high speed data rate ranging from 144 Kbps for mobile users to 2 Mbps for stationary users. It is also able to provide high speed internet service with exceptional data rate for wireless service. CDMA is a spread spectrum multiple access technique. In CDMA a locally generated code runs at a much higher rate than the data to be transmitted. Each user in a CDMA system uses a different code to modulate their signal. Choosing the codes used to modulate the signal is very important in the performance of CDMA systems. The best performance will occur when there is good separation between the signal of a desired user and the signals of other users. The separation of the signals is made by correlating the received signal with the locally generated code of the desired user. If the signal matches the desired user's code then the correlation function will be high and the system can extract that signal. If the desired user's code has nothing in common with the signal the correlation should be as close to zero as possible (thus eliminating the signal); this is referred to as cross correlation. If the code is correlated with the signal at any time offset other than zero, the correlation should be as close to zero as possible. This is referred to as auto-correlation and is used to reject multi-path interference.

In general, CDMA belongs to two basic categories: synchronous (orthogonal codes) and asynchronous (pseudorandom codes). The PN codes are used for the encryption of the data for the security of data which is also known as the data scrambling while the orthogonal codes are used for spreading purpose.

1.3.2 Multiple access

The concept behind multiple accesses is to permit a number of users to share a common channel. The two traditional ways of multiple accesses are Frequency Division Multiple Access (FDMA) and Time Division Multiple Access (TDMA). Code Division Multiple Access (CDMA) is the third and new multiple access technique that does not allocate frequency or time in user slots but gives the right to use both to all users simultaneously. To do this, it uses a technique known as Spread Spectrum.

FDMA

In Frequency Division Multiple Access, the frequency band is divided in slots. Each user gets one frequency slot assigned that is used at will. It could be compared to AM or FM broadcasting radio where each station has a frequency assigned. FDMA demands good filtering.

TDMA

In Time Division Multiple Access, the frequency band is not partitioned but users are allowed to use it only in predefined intervals of time, one at a time. Thus, TDMA demands synchronization among the users.

CDMA

In CDMA each user is assigned a different unique code that propagates the data in the channel irrespective of the frequency or time.

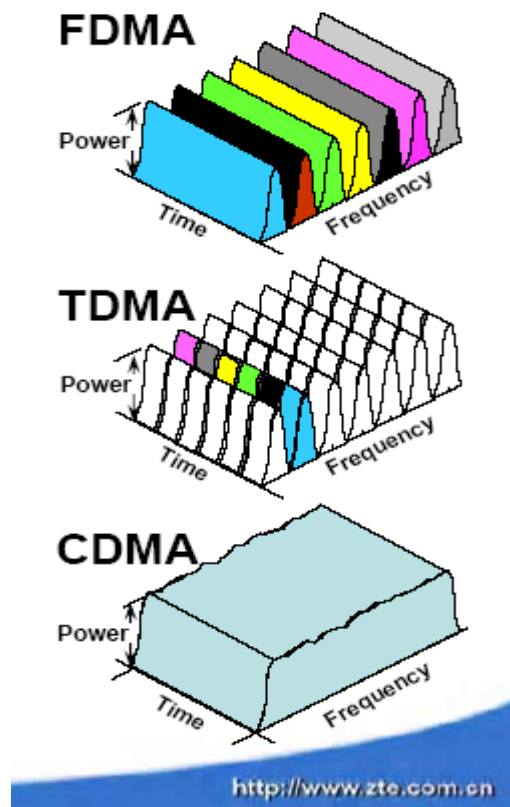


Figure 1.1: Multiple Access Techniques

(Courtesy: ZTE Corporation)

1.3.3 Spread Spectrum types

1. FHSS (Frequency Hopped Spread Spectrum)

Frequency Hopped spread Spectrum involves a periodic change of transmission frequency. A frequency hopping signal may be regarded as a sequence of modulated data bursts with time varying, pseudorandom carrier frequencies. Hopping occurs over a frequency band that includes a number of channels. Data is sent by hopping the transmitter carrier to seemingly random channels that are known only to the desired receiver.

Frequency hopping may be classified as fast or slow. Fast frequency hopping occurs if there is more than one frequency hop during each transmitted symbol. Thus fast frequency hopping implies that the hopping rate equals or exceeds the information symbol rate. Slow frequency hopping occurs if more than one symbol is transmitted in the time interval between frequency hops.

2. DSSS (Direct Sequence Spread Spectrum):

The DSSS system is a wide band system in which the entire bandwidth of the system is available to each user. A system is defined as a DSSS system if it satisfied the following requirements:

- The spreading signal has a bandwidth much larger than the minimum bandwidth required transmitting the desired information for which a digital system is a base band data.
- The spreading of the data is performed by means of a spreading signal often called a code signal. The code signal is independent of the data and of a much higher rate than the data signal.
- At the receiver, de-spreading is accomplished by the cross-correlation of the received spread signal with a synchronized replica of the same signal used to spread the data.

1.3.4 Spreading codes

PN Sequence

In CDMA, the PN sequences are used to

- Spread the bandwidth of the modulated signal to the larger transmission bandwidth.
- Distinguish between the user signals by utilizing the same transmission bandwidth in the multiple access schemes.

PN sequences are not random. They are deterministic periodic sequences. The following are the two key properties of an ideal PN sequences:

- The relative frequencies of 0 and 1 are nearly $\frac{1}{2}$.
- The run lengths of 0s or 1s are: $\frac{1}{2}$ of all run lengths are of length 1; $\frac{1}{4}$ are of length 2; $\frac{1}{8}$ are of length 3; and so on.

The PN sequences are generated by combining the outputs of the feedback registers. A feedback register consists of consecutive two-stage memory or storage stages and feedback logic. Binary registers are shifted through the shift register in response to the clock pulses. They have nearly equal number of 1's and 0's.

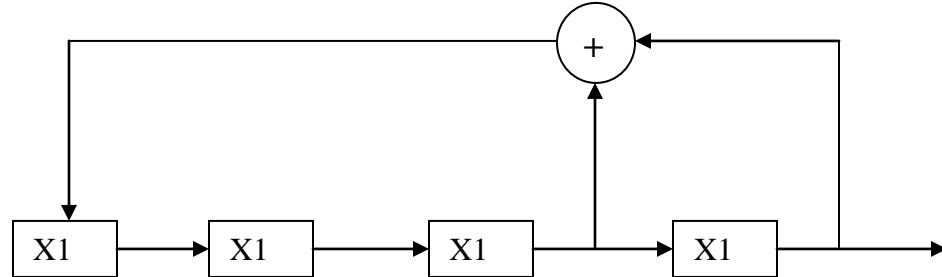


Figure 1.2: Four stage feedback register

Orthogonal codes

Orthogonal functions are employed to improve the bandwidth efficiency of SS system. While there are many different sequences that can be used to generate an orthogonal set of functions, the Walsh and Hadamard sequences make useful sets for CDMA. They have equal number of 1s and 0s. The Hadamard matrix is generated by following methods:

$$H_0 = [0] \quad H_{2N} = \begin{bmatrix} H_n & H_n \\ H_n & \overline{H_n} \end{bmatrix}$$

1.3.5 Auto Correlation and Cross Correlation

Auto Correlation

Auto Correlation is defined as the product of the signal with its time shifted version. For orthogonal codes, the autocorrelation function produces a very high value.

$$R_x(\tau) = \int_{-\infty}^{\infty} x(t)x(t+\tau)dt, \quad \text{for } -\infty < \tau < \infty$$

Cross Correlation

Cross Correlation is defined as the product of the signal with different signal. Cross correlation of any signal with orthogonal sequences produces 0 value. For PN codes, the cross correlation produces nearly 0 value.

$$R_C(\tau) = \int_{-T_o/2}^{T_o/2} x(t)y(t+\tau)dt, \quad \text{for } -\infty < \tau < \infty$$

1.3.6 CDMA operation from speech input to speech output

CDMA Communication Model

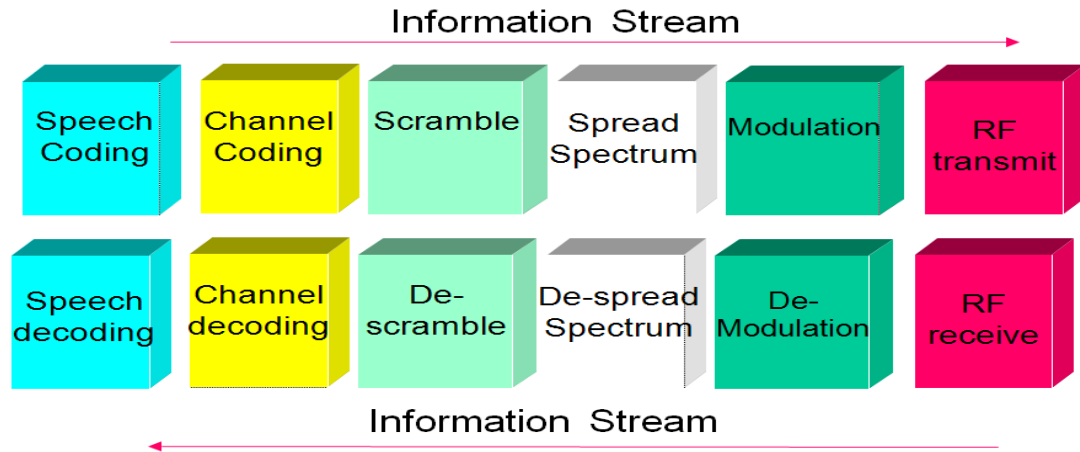


Figure 1.3: CDMA Communication Model (Courtesy: ZTE Corporation)

The following description is made with reference to IS-95 Standard.

Source coding

It involves sampling, quantizing, and digitizing the speech signals. For telephone, the speech frequencies range from 300-3400 Hz. To achieve telephone quality, 12 bit per sample are required at 8000 samples/second. However by using logarithmic sampling system, 8 bit/sample are sufficient. Each sample is then quantized to one of 256 levels. Two wide variations of PCM based on non-uniform quantization are used to achieve quality speech: μ law and A law.

μ law: The basis of μ law is

$$s(t) = \text{sgn}[i(t)] \frac{\ln(1 + \mu(\text{abs}[i(t)]))}{\ln(1 + \mu)}, \quad \text{for } -1 \leq i(t) \leq 1$$

the typical value of $\mu = 255$.

A law: The basic theory of A law encoding is as follows

$$s(t) = \text{sgn}[i(t)] \frac{1 + \ln(A[\text{abs}(i(t))])}{1 + \ln(A)}, \quad \text{for } 1/A \leq \text{abs}[i(t)] \leq 1$$

$$s(t) = \text{sgn}[i(t)] \frac{1 + (A[\text{abs}(i(t))])}{1 + \ln(A)}, \quad \text{for } 0 \leq \text{abs}[i(t)] \leq 1/A$$

Channel Coding:

Channel Coding is used to provide the coding gain that is defined as the reduction in the required E_b/N_0 in dB to achieve a specified error probability of the coded system over an uncoded system with the same modulation and channel characteristics. The channel coding process usually falls into two classes: Block Codes and Convolutional Codes. There are many sub classes of block codes including linear block codes, Binary Cyclic Codes and BCH. IS95 system uses Convolutional codes based on Viterbi algorithm.

Convolutional Codes:

Convolutional Encoders can be thought of as finite state machines that change states as the function of the input sequence. A Convolutional code is generated by passing an information sequence through a finite state shift register. The shift register contains k stages and m linear algebraic function based on generator polynomials. The number of output bits for each k bit input data sequence is n bits. The code rate is $r=k/n$. The parameter k is called the constraint length and indicates the number of input data bits upon which the current output is dependent.

Viterbi Algorithm:

Viterbi Algorithm performs maximum likelihood decoding. It reduces the computational load by taking advantage of the special structure in coder trellis. The complexity of the Viterbi decoder is not a function of number of symbols in the code word sequence. The Viterbi algorithm removes from the consideration those trellis paths that could not possibly be candidates for the maximum likelihood choices. When two paths enter the same state, the one having the best metric is selected; this path is called the surviving path. This selection of surviving path is performed for all the states. The decoder continues in this way to advance deeper into the trellis, making decisions by eliminating the least likely paths. The major drawback of the Viterbi algorithm is that, while error probability decreases exponentially with constraint length, the number of code states and consequently decoder complexity grows exponentially with constraint length.

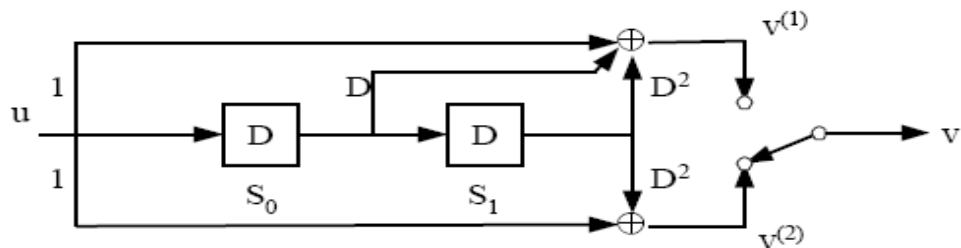


Figure 1.4: Convolutional Encoder

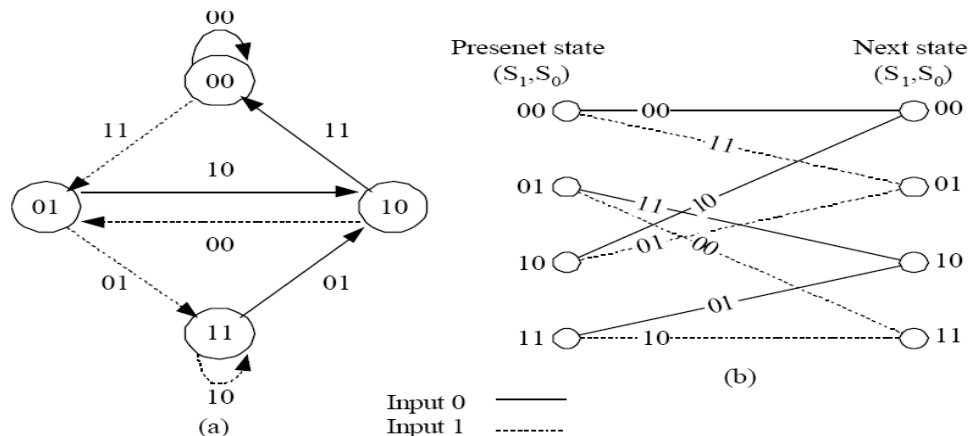


Figure 1.5: (a) State Diagram and (b) trellis diagram of Convolutional Encoder of Figure 1.4

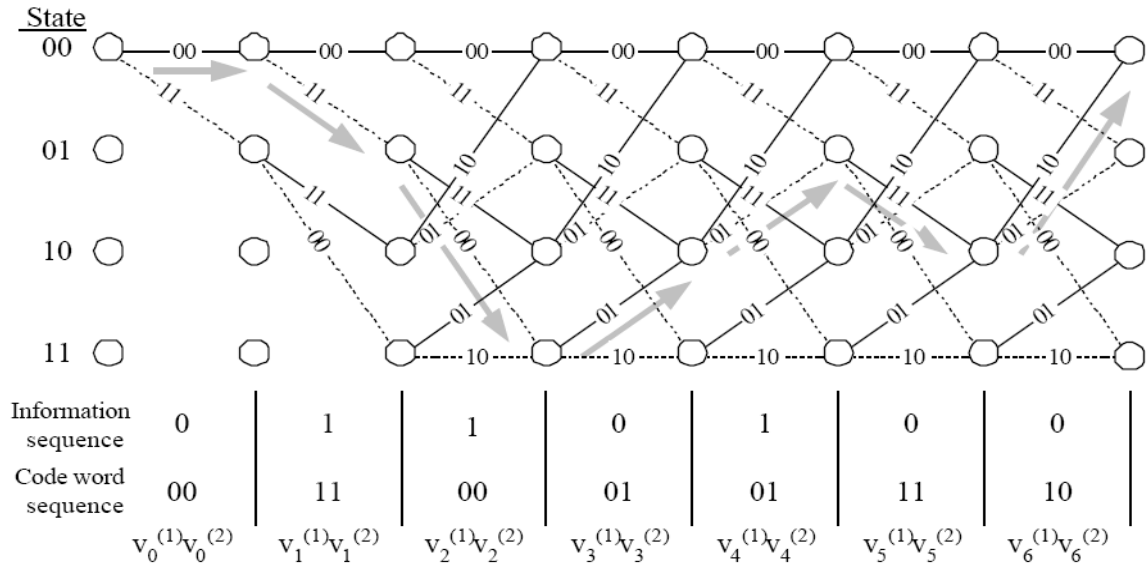


Figure 1.6 An encoding Example for Figure 1.5

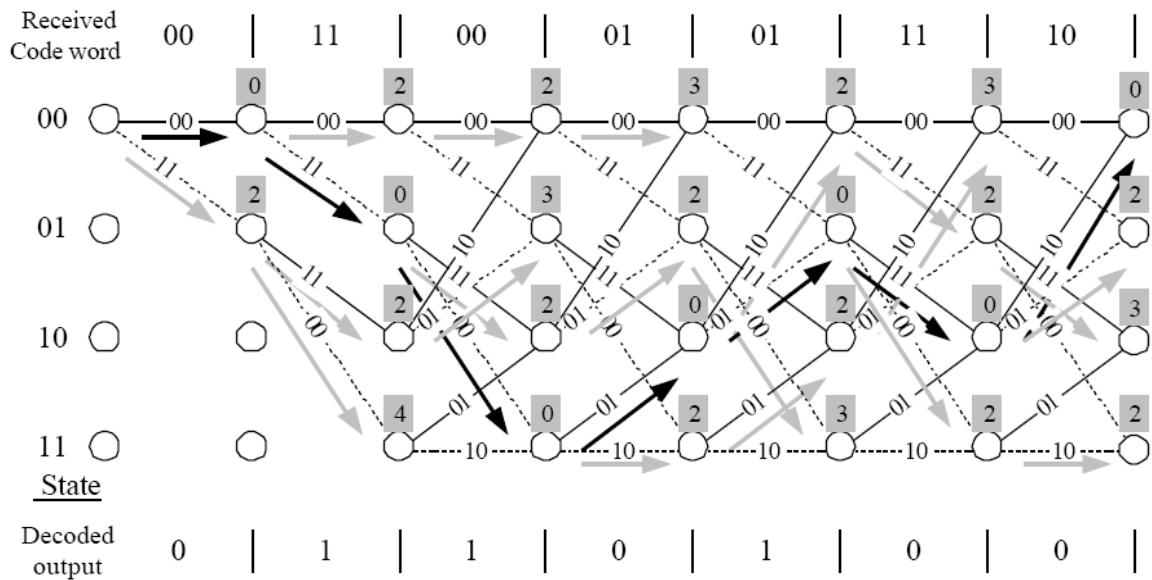


Figure 1.7: Decoded Output of Trellis

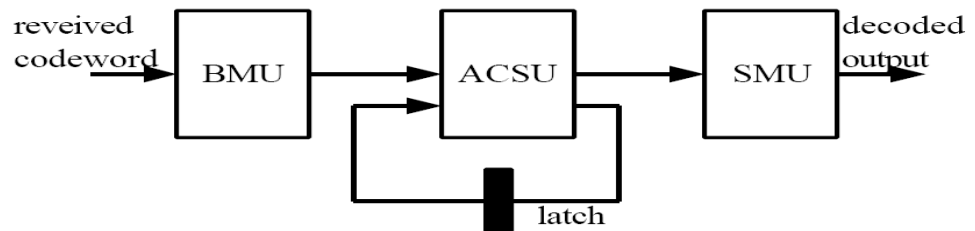


Figure 1.8: Block Diagram of Viterbi Decoder

(Figures 1.4,1.5.1.6,1.7 and 1.8 Courtesy: Paper on FPGA Realization of the Viterbi Decoder for HDSL2 Systems, Feng Lo, National Central University Chung-Li, 32054, Taiwan, ROC)

Interleaving:

Interleaving is used to obtain time diversity in a digital communication system without adding any overhead. Because speech coders represent wide range of voices in uniform and efficient digital format, the encoded data bits carry wide range of information and some are more important than others and must be protected from errors. It is the function of interleaver to spread these bits out in time so that, if there is deep fade or noise burst, the important bits from block are not corrupted at same time. The interleaver is of two types- block interleaver and convolution interleaver.

A block interleaver formats the encoded data into rectangular array of m rows and n columns, and interleaves mn bits at a time. At the receiver, the de-interleaver stores the received data sequentially increases the row number of each successive bit, and then clocks out the data row-wise.

The convolution interleaver is ideally suited for convolution encoder.

Scrambling (Encryption):

Scrambling is performed after the block interleaver. The data scrambling is performed by modulo-2 addition of the interleaved data with the Long PN sequence whose bits are masked with the ESN (Electronic Serial Number).

Spreading:

The spreading of the traffic data is done by the Walsh function which comprises of 64 binary sequences each of length 64 which are completely orthogonal to each other and provide orthogonal channelization for all users. A user that is spread using Walsh function n is assigned channel number n (n=0 to n=64). In each user, the data symbol is spread by 64 Walsh chips. For commercial CDMA, the 64 X 64 Walsh function is generated by the following recursive procedure.

$$H_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad H_2 = \begin{bmatrix} 00 \\ 01 \end{bmatrix} \quad H_4 = \begin{bmatrix} 0000 \\ 0101 \\ 0011 \\ 0110 \end{bmatrix} \quad H_{2N} = \begin{bmatrix} H_n & H_n \\ H_n & \overline{H_n} \end{bmatrix}$$

The product of each row elements with the corresponding row elements of any other row is 0. This process is also known as encoding process. This is a very useful feature that makes the transmission of the data reliable and efficient compared to other access methods.

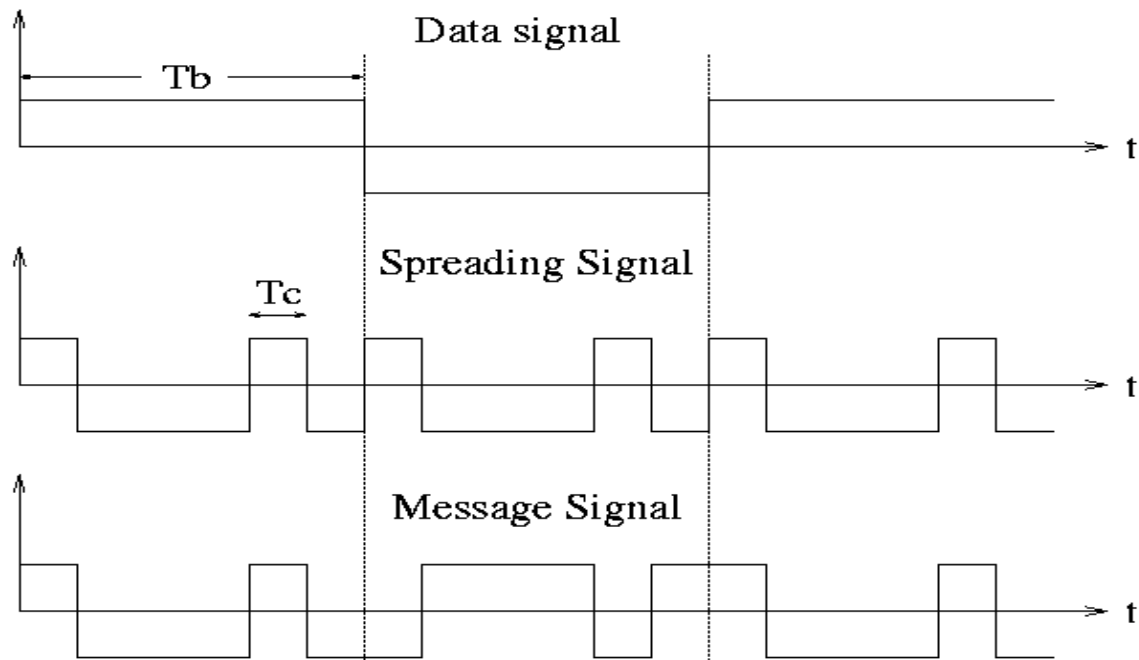


Figure 1.9: Spreading Process

Despreading

Despreading is the process of decoding the spreaded data. Despreading of the data is done in the receiver side so that the actual user data can be obtained from the spread data. This is done by multiplying the spread data by the same orthogonal codes. The output from this process is treated in such a way that any data greater than 0 is treated as '1' while any data less than 0 is treated as '0'. When the spreaded data is correlated by any other Walsh codes, the receiver experiences noise. Hence it is clear that the sender did not transmit any data to that user.

Spreading and de-spreading principle for two users

If sender0 has code (1,-1) and data (1,0,1,1), and sender1 has code (1,1) and data (0,0,1,1), and both senders transmit simultaneously, then this table describes the coding steps:

Step	Encode sender0	Encode sender1
0	vector0=(1,-1), data0=(1,0,1,1)=(1,-1,1,1)	vector1=(1,1), data1=(0,0,1,1)=(-1,-1,1,1)
1	encode0=vector0.data0	encode1=vector1.data1
2	encode0=(1,-1).(1,-1,1,1)	encode1=(1,1).(-1,-1,1,1)
3	encode0=((1,-1),(-1,1),(1,-1),(1,-1))	encode1=((-1,-1),(-1,-1),(1,1),(1,1))
4	signal0=(1,-1,-1,1,1,-1,1,-1)	signal1=(-1,-1,-1,-1,1,1,1,1)

Because signal0 and signal1 are transmitted at the same time into the air, they add to produce the raw signal:

$$(1,-1,-1,1,1,-1,1,-1) + (-1,-1,-1,-1,1,1,1,1) = (0,-2,-2,0,2,0,2,0)$$

This raw signal is called an interference pattern. The receiver then extracts an intelligible signal for any known sender by combining the sender's code with the interference pattern, the receiver combines it with the codes of the senders. The following table explains how this works and shows that the signals do not interfere with one another:

Step	Decode sender0	Decode sender1
0	vector0=(1,-1), pattern=(0,-2,-2,0,2,0,2,0)	vector1=(1,1), pattern=(0,-2,-2,0,2,0,2,0)
1	decode0=pattern.vector0	decode1=pattern.vector1
2	decode0=((0,-2),(-2,0),(2,0),(2,0)).(1,-1)	decode1=((0,-2),(-2,0),(2,0),(2,0)).(1,1)
3	decode0=((0+2),(-2+0),(2+0),(2+0))	decode1=((0-2),(-2+0),(2+0),(2+0))
4	data0=(2,-2,2,2)=(1,0,1,1)	data1=(-2,-2,2,2)=(0,0,1,1)

(Courtesy: Spreading and De-Spreading Principle for two users:
<http://www.wikipedia.org>)

Modulation

Modulation is defined as the process of changing the characteristics of the carrier signal with the baseband signal so that the baseband signal can be transmitted wirelessly. The modulation technique applied in the CDMA is Quadrature Phase Shift Keying (QPSK) modulation. The advantage of QPSK is that it has twice the bandwidth efficiency of BPSK and its phase can represent 1 of 4 symbols.

Demodulation

Demodulation involves retrieving the baseband signal from the modulated signal for further signal processing. The QPSK demodulator applies the in-phase and quadrature phase correlation to get the baseband digital signals.

1.3.7 Computer Interface

Parallel Interface:

The Parallel Port is the most commonly used port. The most prominent advantage of the parallel port interfacing is the capability of the transfer of the 8 bit data simultaneously but however suffers from the distance limitations.

The port is composed of 4 control lines, 5 status lines and 8 data lines. It's found commonly on the back of PC as a D-Type 25 Pin female connector.

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out PaperEnd	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer nSelect-In	In/Out	Control	Yes
18 - 25	19-30	Ground	Gnd		

Table 1.1: Pin Assignments of D-Type 25 pin Parallel Port Connector.

Serial Interface

Serial interfacing is another way of communicating with the PC. In serial data communications, the individual data bits are transferred one by one in a serial manner via a single data line. For doing so the data is first split into bits and the transfer is done bit by bit.

Advantages of serial data transfer

- Serial Cables can be longer than Parallel cables
- A single wire is used for data transmission. If a device needs to be mounted a far distance away from the computer then 3 core cable (Null Modem Configuration) is going to be a lot cheaper than running 19 or 25 core cable.

Serial data transfer can be simplex, half-duplex or full-duplex. A **simplex** data line can transmit data only in one direction. In **half-duplex**, communication can take place in either direction between two systems, but can occur in only one direction at a time.

However in **full-duplex** data transfer is possible in both directions between transmitter and the receiver simultaneously.

Serial data can be sent synchronously or the asynchronously. In **synchronous transfer** of serial data, one or more additional signals are transmitted to indicate when the next bit is valid on the data line. These signals may be formed by a clock signals of a clock signal source or by handshake signals of the form request and acknowledge. For **asynchronous transmission**, each data character has a bit which identifies its start and 1 or 2 bits which identify its end. Since each character is individually identified, characters can be sent at any time (asynchronously) in a same way a person types in a keyboard.

Serial data unit

The figure below shows the bit format used for transmitting asynchronous serial data.

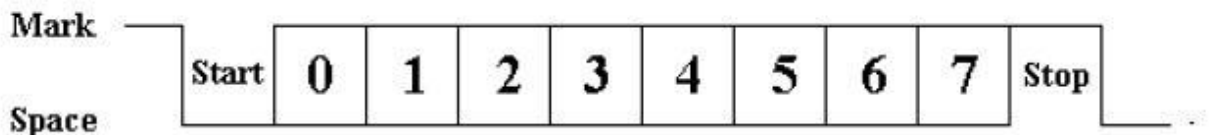


Figure 1.10: Bit format used for sending asynchronous serial data

When no data is being sent, the signal line is in a constant high or **mark** state. The beginning of a data character is indicated by the line going low for one bit time. This bit is called a **start bit**. The data bits are then sent out on the line one after the other. Note that the least significant bit is sent out first. Following the data bit is a parity bit which is used to check for errors in received data. After then the signal line is raised high for at least one bit time to identify the end of a character. This always high bit is referred to as a **stop bit**.

Serial Ports come in two "sizes", There are the D-Type 25 pin connector and the D-Type 9 pin connector both of which are male on the back of the PC.

D-Type-25 Pin No.	D-Type-9 Pin No.	Abbreviation	Full Name
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request To Send
Pin 5	Pin 8	CTS	Clear To Send

Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator

Table 1.2: Pin Connection of Serial Port

1.3.8 Asynchronous Serial Communication:

UART

Universal asynchronous receiver and transmitter (UART) is a circuit that sends parallel data through a serial line. Because the voltage level defined in RS-232 is different from that of FPGA I/O, a voltage converter chip is needed between a serial port and a FPGAs I/O pins

A UART includes a transmitter and a receiver. The transmitter is essentially a special shift register that loads data in parallel and then shifts it out bit by bit at a specific rate. The receiver, on the other hand, shifts in data bit by bit and then reassembles the data. The serial line is '1' when it is idle. The transmission starts with a start bit, which is '0', followed by data bits and an optional parity bit, and ends with stop bits, which is '1'. The number of data bits can be 6, 7, or 8. The optional parity bit is used for error detection. For odd parity, it is set to '0' when the data bits have an odd number of 1's. For even parity, it is set to '0' when the data bits have an even number of 1's. The number of stop bits can be 1, 1.5, or 2.

The transmission with 8 data bits, no parity, and 1 stop bit is shown in Figure 7.1. Note that the LSB of the data word is transmitted first. No clock information is conveyed through the serial line. Before the transmission starts, the transmitter and receiver must agree on a set of parameters in advance, which include the baud rate (i.e., number of bits per second), the number of data bits and stop bits, and use of the parity bit. The commonly used baud rates are 2400, 4800, 9600, and 19,200 bauds.

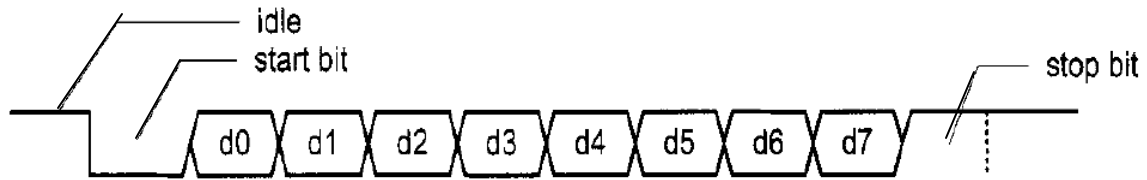


Figure 1.11: Transmission of a byte of serial data

UART Receiving Subsystem

Since no clock information is conveyed from the transmitted signal, the receiver can retrieve the data bits only by using the predetermined parameters. The oversampling scheme is used to estimate the middle points of transmitted bits and then retrieve them at these points accordingly.

Oversampling procedure

The most commonly used sampling rate is 16 times the baud rate, which means that each serial bit is sampled 16 times. Assume that the communication uses N data bits and M stop bits. The oversampling scheme works as follows:

1. Wait until the incoming signal becomes '0', the beginning of the start bit, and then start the sampling tick counter.
2. When the counter reaches 7, the incoming signal reaches the middle point of the start bit. Clear the counter to 0 and restart.
3. When the counter reaches 15, the incoming signal progresses for one bit and reaches the middle of the first data bit. Retrieve its value, shift it into a register, and restart the counter.
4. Repeat step 3 $N-1$ more times to retrieve the remaining data bits.
5. If the optional parity bit is used, repeat step 3 one time to obtain the parity bit.
6. Repeat step 3 M more times to obtain the stop bits.

The oversampling scheme basically performs the function of a clock signal. Instead of using the rising edge to indicate when the input signal is valid, it utilizes sampling ticks to estimate the middle point of each bit. While the receiver has no information about the exact onset time of the start bit, the estimation can be off by at most $1/16$. The subsequent data bit retrievals are off by at most $1/16$ from the middle point as well. Because of the oversampling, the baud rate can only be a small fraction of the system clock rate, and thus this scheme is not appropriate for a high data rate.

The conceptual block diagram of a UART receiving subsystem is shown in Figure It consists of three major components:

- **UART receiver:** the circuit to obtain the data word via oversampling
- **Baud rate generator:** the circuit to generate the sampling ticks
- **D Flip Flop :** the latch that provides buffer and status between the UART receiver and the system that uses the UART

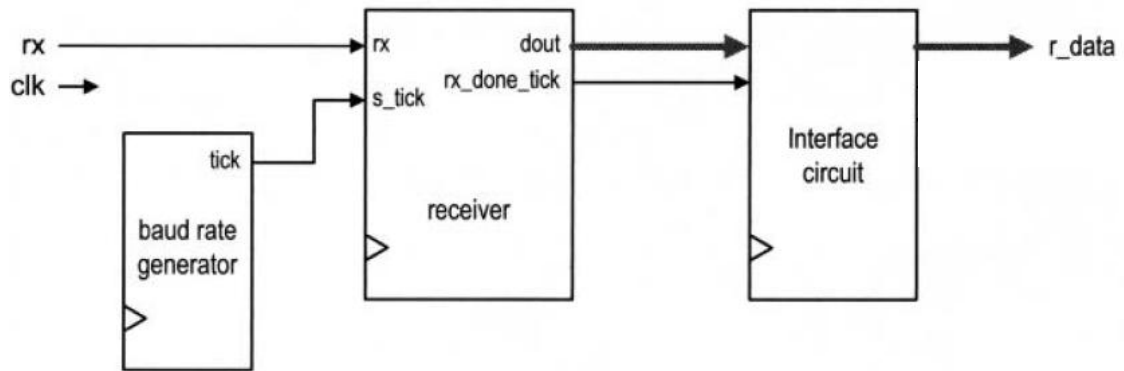


Figure 1.12: UART Receiving Subsystem

(Courtesy: FPGA PROTOTYPING BY VHDL EXAMPLES,
Pong P. Chu, Cleveland State University)

MAX232

Since the RS232 is not compatible with today's microprocessors and microcontroller, a line driver (voltage converter) is needed to convert the RS232's signals to TTL voltage levels that will be acceptable to the FPGA's input pins. One example of such a converter is MAX232 from Maxin Corp. The MAX232 converts from RS232 voltage levels to TTL voltage levels, and vice versa. One advantage of the MAX232 chip is that it uses a +5V power supply which, is the same as the source voltage for the FPGA. The MAX232 has two sets of line drivers for transferring and receiving data. MAX232 require four capacitors ranging from 1 to 22uf.

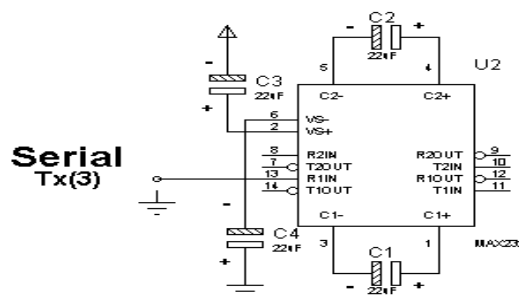


Figure 1.13: MAX232 Pin Layout

1.3.9 LCD Interfacing

LCD

V_{CC} , V_{SS} and V_{EE}

While V_{CC} and V_{SS} provide +5V and ground, respectively, V_{EE} is used for controlling LCD contrast.

RS, register select

If the RS pin=0 the instruction command code register is selected, allowing the user to send command. If it is 1, it is used to send data on the LCD.

R/W register

When R/W=0, writing is enabled. When R/W=1, reading information from it is enabled.

EN (enable) register

The enable pin is used by the LCD to latch the information presented to its data pins. When the data is supplied to the data pins, the high to low must be applied to this pin in order for the pins to latch the data present at the data pins. This pulse must be minimum 450ns wide.

D0-D7

The 8 bit data pins d0-d7 are used to send information to the LCD or read the content of LCD'S internal register. To display the letters and numbers, the ASCII codes are sent for the letters A-Z, a-z, and the numbers 0-9 to these pins while making RS=1. There are also instruction codes that can be sent to the LCD to clear the display or force the cursor to the home position or blink the cursor.

The RS=0 is used to check the busy flag bit to see if the LCD is ready to receive information. The busy flag is d7 and can be read when RS=0 and R/W=1. If R/W=1, and RS=0, when the d7=1 (busy flag =1), the LCD display is busy taking care of internal operation and will not accept any new information. When d7=0, the LCD is ready to receive the new information. It is recommended to check the busy flag before writing any data to the LCD.

Pin	Symbol	I/O	Description
1	V _{SS}	--	Ground
2	V _{CC}	--	+5V power supply
3	V _{EE}	--	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB0	I/O	The 8-bit data bus
9	DB0	I/O	The 8-bit data bus
10	DB0	I/O	The 8-bit data bus
11	DB0	I/O	The 8-bit data bus
12	DB0	I/O	The 8-bit data bus
13	DB0	I/O	The 8-bit data bus
14	DB0	I/O	The 8-bit data bus
15	LED +	--	+5 power supply for Backlight
16	LED -	--	Ground for Backlight

Table 1.3: Pin Description of 16*2 line LCD

1.3.10 Modern Digital Design using HDL

Digital Design Methodology using Hardware Description Language

A digital system can be described at different levels of abstraction and from different points of view. Hardware Description Language(HDL) should accurately model and describe a circuit, whether already built or under development, from either the structural or behavioral views, at the desired level of abstraction. Because HDLs are modeled after hardware, their semantics and use are very different from those of traditional programming languages.

In most of the traditional general-purpose programming languages, such as C,C++,etc operations are performed in sequential order, one operation at a time. Since an operation frequently depends on the result of an earlier operation, the order of execution cannot be altered at will. The sequential process model has two major benefits. At the abstract level, it helps the human thinking process to develop an algorithm step by step. At the implementation level, the sequential process resembles the operation of a basic computer model and thus allows efficient translation from an algorithm to machine instructions.

The characteristics of digital hardware, on the other hand, are very different from those of the sequential model. A typical digital system is normally built by smaller parts, with customized wiring that connects the input and output ports of these parts. When a signal changes, the parts connected to the signal are activated and a set of new operations is initiated accordingly. These operations are performed concurrently, and each operation will take a specific amount of time, which represents the propagation delay of a particular part, to complete. After completion, each part updates the value of the corresponding output port. If the value is changed, the output signal will in turn activate all the connected parts and initiate another round of operations. This description shows several unique characteristics of digital systems, including the connections of parts, concurrent operations, and the concept of propagation delay and timing. The sequential model used in traditional programming languages cannot capture the characteristics of digital hardware, and there is a need for special languages (i.e., HDLs) that are designed to model digital hardware.

HDL's are used to describe the architecture and behavior of discrete electronic systems. HDL's were developed to deal with increasingly complex designs. Top- down, HDL based system design is most useful in large projects, where several designers or teams of designers are working concurrently. HDL's provide structured development. After major architectural decisions have been made, and major components and their connections have been identified, work can proceed independently on subprojects.

VHDL and Verilog are examples of most commonly used HDL languages.

Features of modern HDL language

- The language semantics encapsulate the concepts of entity, connectivity, concurrency, and timing.
- The language can effectively incorporate propagation delay and timing information.
- The language consists of constructs that can explicitly express the structural implementation (i.e., a block diagram) of a circuit.
- The language incorporates constructs that can describe the behavior of a circuit, including constructs that resemble the sequential process of traditional languages, to facilitate abstract behavioral description.
- The language can efficiently describe the operations and structures at the gate and RTL levels.
- The language consists of constructs to support a hierarchical design process.

Use of an HDL program

The HDL program plays three major roles:

1. **Formal documentation.** A digital system normally starts with a word description. Since human language is not precise, the description is frequently incomplete and ambiguous, and the same description may be subject to different interpretations. Because the semantics and syntax of an HDL are defined rigorously, a system specified in an HDL program is explicit and precise. Thus, an HDL program can be used as a formal system specification and documentation among various designers and users.
2. **Input to a simulator.** Simulation is used to study and verify the operation of a circuit without constructing the system physically. An HDL simulator provides a framework to model the concurrent operations in a sequential host computer, and has specific knowledge of the language's syntactic constructs and the associated semantics. An HDL program, combined with test vector generation and a data collection code, forms a test bench, which becomes the input to the HDL simulator. During execution, the simulator interprets HDL code and generates responses accordingly.
3. **Input to a synthesizer.** The modern development flow is based on the refinement process, which gradually converts a high-level behavioral description to a low-level structural description. Some refinement steps can be performed by synthesis software. The synthesis software takes an HDL program as its input and realizes the circuit from the components of a given library. The output of the synthesizer is a new HDL program that represents the structural description of the synthesized circuit.

Advantages of HDLs

HDLs has several fundamental advantages over a traditional gate-level design methodology. Among the advantages are the following:

- The verification of design functionality can be made early in the design process, and immediately simulate a design written as an HDL description. Design simulation at this higher level, before implementation at the gate-level, allows testing architectural and designing decisions.
- FPGA Express provides logic synthesis and optimization, so automatic conversion of a VHDL description to a gate-level implementation can be made in a given technology. This reduces circuit design time and errors introduced when hand-translating a VHDL specification to gates. With FPGA Express logic optimization, automatic transformation of a synthesized design to a smaller and faster circuit can be made.
- HDL descriptions provide technology-independent documentation of a design and its functionality. An HDL description is more easily read and understood than a netlist or schematic description. Since the initial HDL design description is technology-independent, later it can be reused to generate the design in a different technology, without having to translate from the original technology.
- HDL's like VHDL, provides strong type checking like most high-level software languages. A component that expects a four-bit-wide signal type cannot be connected to a three- or five-bit-wide signal; this mismatch causes an error when the HDL description is compiled. If a variable's range is defined as 1 to 15, an error results from assigning it a value of 0. Incorrect use of types has been shown to be a major source of errors in descriptions. Type checking catches this kind of error in the HDL description even before a design is generated.

Description of VHDL

VHDL is one of just a few HDLs in widespread use today. VHDL is recognized as a standard HDL by the IEEE (IEEE Standard 1076, ratified in 1987) and by the United States Department of Defense (MIL-STD-454L). VHDL divides entities (components, circuits, or systems) into an external or visible part (entity name and connections) and an internal or hidden part (entity algorithm and implementation). After defining the external interface to an entity, other entities can use that entity when they all are being developed. This concept of internal and external views is central to a VHDL view of system design. An entity is defined, with respect to other entities, by its connections and behavior. Exploration of alternate implementations (architectures) of an entity without changing the rest of the design can be done. After the entity is defined for one design, reuse of it can be made in other designs as needed. Libraries of entities can be developed for use by many designs, or for a family of designs.

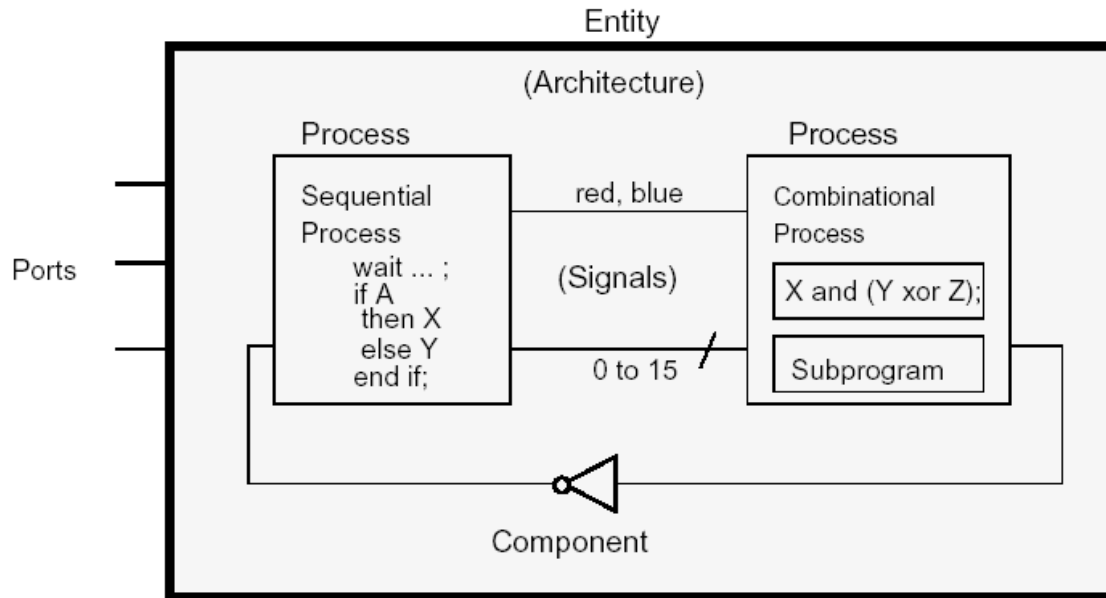


Figure 1.14: VHDL Hardware Model

A VHDL entity (design) has one or more input, output, or inout ports that are connected (wired) to neighboring systems. An entity is itself composed of interconnected entities, processes, and components, all which operate concurrently. Each entity is defined by a particular architecture, which is composed of VHDL constructs such as arithmetic, signal assignment, or component instantiation statements. In VHDL, independent processes model sequential (clocked) circuits, using flip-flops and latches, and combinational (unclocked) circuits, using only logic gates. Processes can define and call (instantiate) subprograms (sub designs). Processes communicate with each other by signals (wires). A signal has a source (driver), one or more destinations (receivers), and a user-defined type, such as “color” or “number between 0 and 15.” VHDL provides a broad set of constructs. With VHDL discrete electronic systems can be described by varying complexity (systems, boards, chips, modules) with varying levels of abstraction. VHDL language constructs are divided into three categories by their level of abstraction: behavioral, dataflow, and structural. These categories are described as follows:

Behavioral

The functional aspects of a design, expressed in a sequential VHDL process.

Dataflow

The view of data as flowing through a design, from input to output. An operation is defined in terms of a collection of data transformations, expressed as concurrent statements.

Structural

The view closest to hardware; a model where the components of a design are interconnected. This view is expressed by component instantiations.

1.3.11 Spartan II FPGA (XC2s50tq144-5)

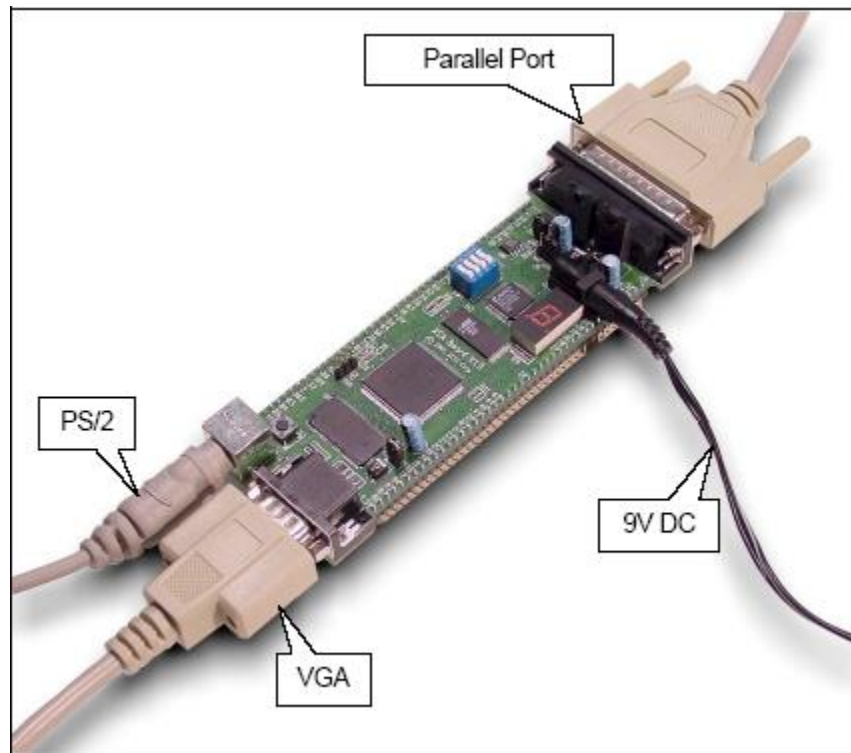


Figure 1.15: External connections to the XSA Board
(Courtesy: XSA Board V1.1, V1.2 User Manual, XESS Corporation)

XSA board Organisation

The XSA Board contains the following components:

- **XC2S50 or XC2S100 Spartan-II FPGA:** This is the main repository of programmable logic on the XSA Board.
- **XC9572XL CPLD:** This CPLD manages the interface between the PC parallel port and the rest of the XSA Board.
- **Oscillator:** A programmable oscillator generates the master clock for the XSA Board.
- **Flash:** A 128 or 256-KByte Flash device provides non-volatile storage for data and configuration bitstreams.
- **SDRAM:** An 8 or 16-MByte SDRAM provides volatile data storage accessible by the FPGA.
- **LED:** A seven-segment LED allows visible feedback as the XSA Board operates.
- **DIP switch:** A four-position DIP switch passes settings to the XSA Board or controls the upper address bits of the Flash device.
- **Pushbutton:** A single pushbutton sends momentary contact information to the FPGA.
- **Parallel Port:** This is the main interface for passing configuration bitstreams and data to and from the XSA Board.

- **PS/2 Port:** A keyboard or mouse can interface to the XSA Board through this port.
- **VGA Port:** The XSA Board can send signals to display graphics on a VGA monitor through this port.

FPGA Programming

The Implementation of a logic design with an FPGA usually consists of the following steps:

1. First the description of the logic circuit is entered using a hardware description language (HDL) . The design can be drawn using a schematic editor.
2. A logic synthesizer is used program to transform the HDL or schematic into a netlist. The netlist is just a description of the various logic gates in the design and how they are interconnected.
3. The implementation tools are used to map the logic gates and interconnections into the FPGA. The FPGA consists of many configurable logic blocks, which can be further decomposed into look-up tables that perform logic operations. The CLBs and LUTs are interwoven with various routing resources. The mapping tool collects the netlist gates into groups that fit into the LUTs and then the place & route tool assigns the groups to specific CLBs while opening or closing the switches in the routing matrices to connect them together.
4. After the completion of the implementation, a program extracts the state of the switches in the routing matrices and generates a bit stream where the ones and zeroes correspond to open or closed switches.
5. The bitstream is downloaded into a physical FPGA chip using tools like IMPACT, XS-Tools, etc. The electronic switches in the FPGA open or close in response to the binary bits in the bitstream. Upon completion of the downloading, the FPGA will perform the operations specified by HDL code or schematic.

XILINX ISE provides the HDL and schematic editors, logic synthesizer, fitter, and bitstream generator software. The XSTOOLS provide utilities for downloading the bitstream into the FPGA on the XSA Board.

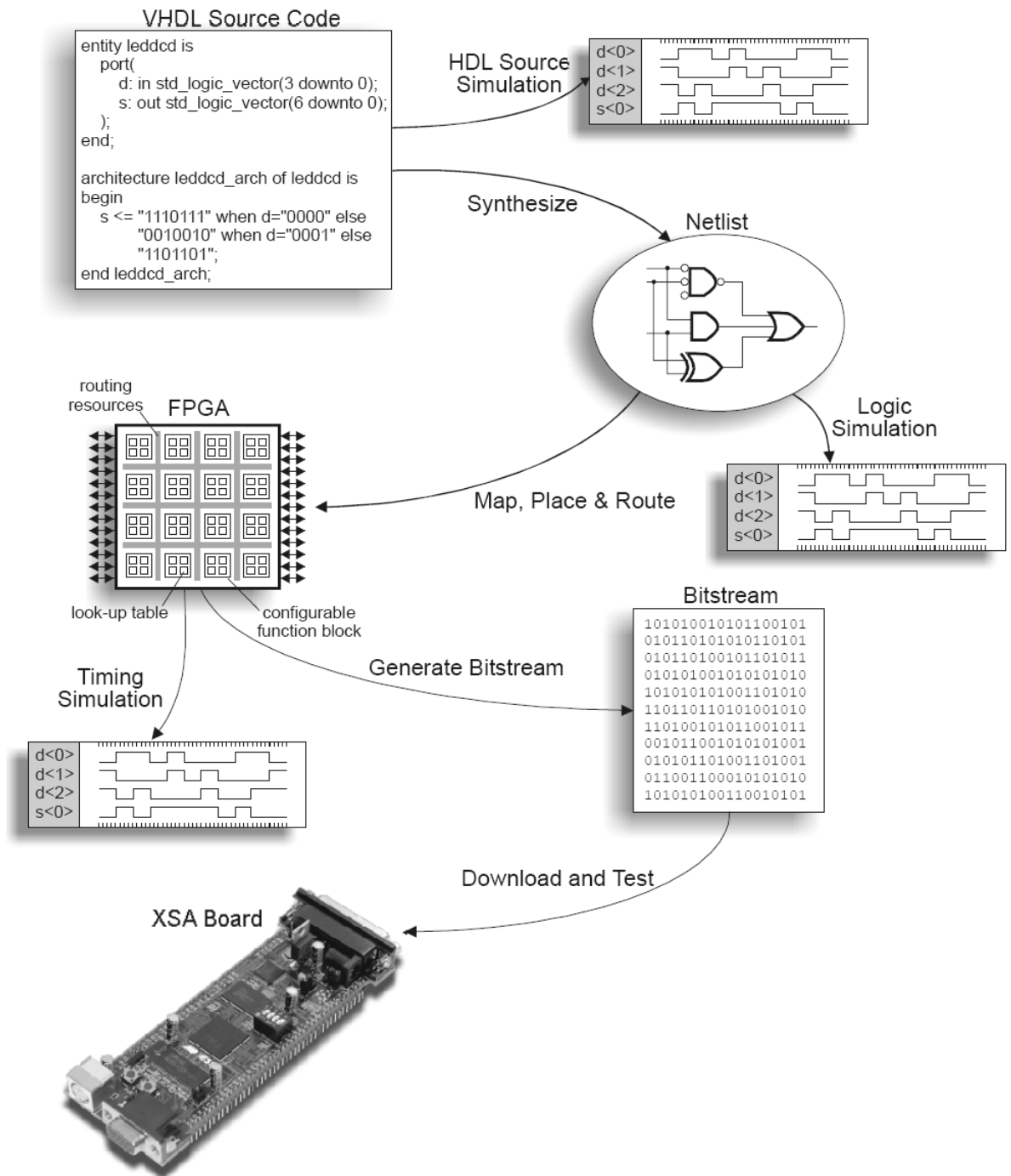


Figure 1.16: FPGA Programming Process
(Courtesy: Xilinx ISE 10 Tutorial, XESS Corporation)

Chapter 2: Implementation Tools

2.1 List of Tools and their brief utility

2.1.1 XC2s50 Spartan-II FPGA:

Field-programmable gate array (FPGA) is a semiconductor device that can be configured by the customer or designer after manufacturing—hence the name "field-programmable". FPGAs are programmed using a logic circuit diagram or a source code in a hardware description language (HDL) to specify how the chip will work. They can be used to implement any logical function that an application-specific integrated circuit (ASIC) could perform, but the ability to update the functionality after shipping offers advantages for many applications.

In the project, the spartan II FPGA XC2s50 is used as both transmitting and receiving unit. One FPGA consists of all required modules for two transmitters plus a common channel and the other FPGA consists of all modules required for two receivers plus the display device.

2.1.2 HDL Language: VHDL:

HDLs are standard text-based expressions of the spatial and temporal structure and behaviour of electronic systems. In contrast to a software programming language, HDL syntax and semantics include explicit notations for expressing time and concurrency, which are the primary attributes of hardware. Languages whose only characteristic is to express circuit connectivity between a hierarchy of blocks are properly classified as netlist languages used on electric computer-aided design (CAD).

VHDL (VHSIC hardware description language) is commonly used as a design-entry language for field-programmable gate arrays and application-specific integrated circuits in electronic design automation of digital circuits.

All the packages and modules required in the project were written in VHDL language. It is also a common HDL language and so is accepted by both modelsim simulator and ISE project navigator.

2.1.3 HDL Simulator: ModelSim XE 6.1c

ModelSim provides a comprehensive simulation and debug environment for complex ASIC and FPGA designs. Support is provided for multiple languages including Verilog, SystemVerilog, VHDL and SystemC.

Each module written in ISE project navigator is tested in ModelSim environment for verification of the data and timing. After each module is tested, all the files are combined in a top hierarchy and simulate the overall project. ModelSim simulation for each module component is required before testing into real hardware. This also provides step-by-step debugging facility.

2.1.4 Synthesis Tool for FPGA: Xilinx ISE Project Navigator

The Xilinx ISE system is an integrated design environment that consists of a set of programs to create (capture), simulate and implement digital designs in a FPGA or CPLD target device. All the tools use a graphical user interface (GUI) that allows all programs to be executed from toolbars, menus or icons.

The behavior of each module is described in ISE project navigator, simulate each module in ModelSim and accumulate all the modules into the top hierarchy. Once the top hierarchy is made, this hierarchy is synthesized, implemented and finally the bit file is generated for given constraints. The bit file then can be loaded into the real hardware using this same software.

2.1.5 Simulation Software: MATLAB

MATLAB is a numerical computing environment and programming language. Maintained by The MathWorks, MATLAB allows easy matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages. Although it is numeric only, an optional toolbox uses the MuPAD symbolic engine, allowing access to computer algebra capabilities. An additional package, Simulink, adds graphical multidomain simulation and Model-Based Design for dynamic and embedded systems.

In the project, the GUI interface of MATLAB is used. Features such as sliders, textbox and axes are utilized for proper data acquisition and visualization. GUI provides data to both simulation and the hardware. Since being user interactive, the data can be given in real time and the real-time processing is done in both software and hardware to verify the results.

Chapter 3: Hardware Design and Emulation

3.1 Design Overview

The transmitter and receiver has been implemented in two FPGA to make more realistic communication. The communication between the transmitter and the receiver is synchronous.

3.2 Design of Communication Block-Sets targeted to FPGA

3.2.1 Transmitter

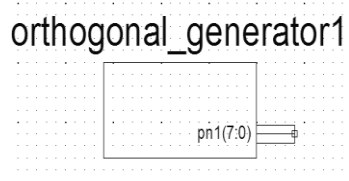
The transmitting FPGA receives user data for two users from the PC. For this the PC's parallel and serial port is used. The Matlab application built is designed such that it provides the data for a user at the parallel port and for another at the serial port. The serial data is received in the FPGA by the UART in the FPGA. The output of UART is a byte of data and the synchronizing signal. The resultant data is then converted to bit stream by parallel to serial converter. The stream of bits then enters the spreading module. The spreading module takes input as 8 bit orthogonal sequence and a bit from the parallel to serial converter and spreads the bit of data according to spread spectrum principle. The orthogonal codes for each user are pre-specified. The spreading process is performed for each user. The spreaded data is then combined in an adder module which just adds the spreaded data bit by bit. The sum which is 8 bit integer array is then fed to parallel to serial converter for transmission in the medium (Wire line Cables). Each module is synchronized asynchronously.

The components used in transmitter can be listed as follows-

- Orthogonal Code Generator
- UART
- Parallel to serial Converter
- Spreading Block

Orthogonal Code Generator:

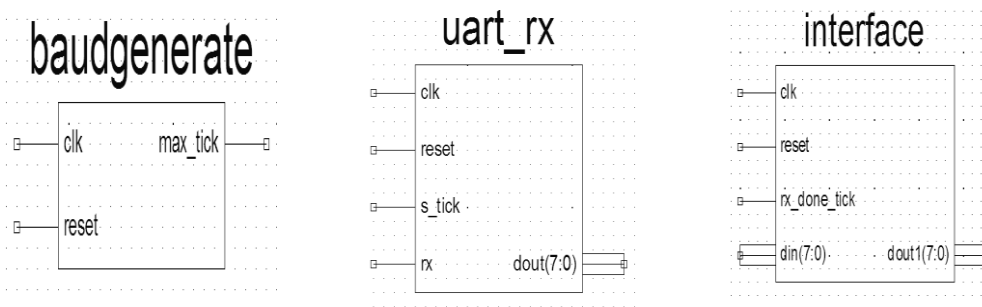
The Orthogonal code generator provides pre-specified 8 bit Orthogonal code generated using 8*8 Hadamard matrix.



UART receiver module:

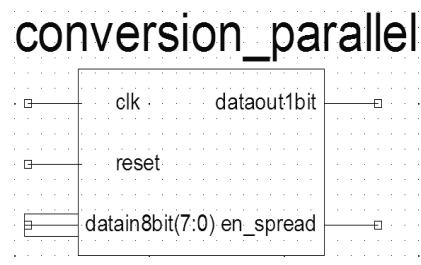
The UART receiver module is composed of three components baud rate generator, UART receiver and the interface circuit.

The baud rate generator generates baud at the rate of 19200 Bd equal to that of PC's serial port. The UART receiver receives serial data from the PC's serial port and baud from baud generator. This module extracts the 8 bit data from the serial data unit of PC's serial data. The interface unit stores the last received 8 bit serial data from the PC which corresponds to a user data.



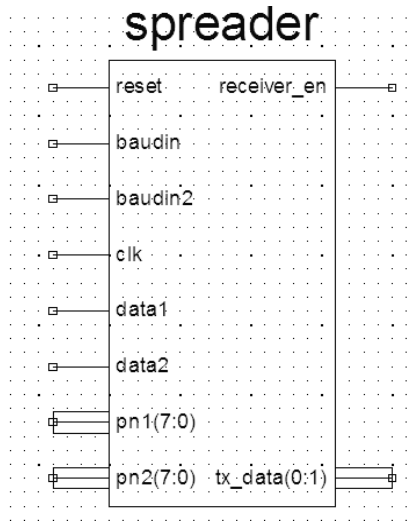
Parallel to serial converter

Two parallel to serial converter are used, the first one is used to convert each user's 8 bit data to serial stream of bits to be fed to spreading block and the other for transmitting the added spreaded data (integer format) to the transmission medium.



Spreading Block

The spreading block accepts input as bit stream and corresponding users 8 bit orthogonal sequence and spreads the corresponding input data and adds the spreaded data. Finally the added data is converted to serial form and transmitted.



3.2.2 Receiver

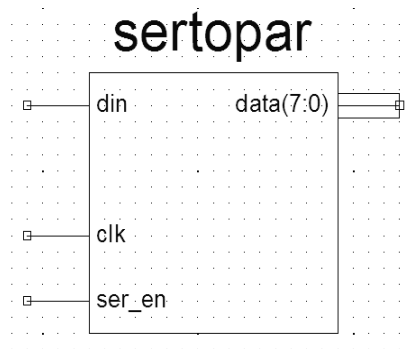
The receiver performs the reverse of the operations performed in the transmitter. The receiver receives the serial data, and converts this to 8 bit parallel data using serial to parallel conversion module. The 8 bit data is then fed to Despreading Block. The input to the Despreading block is the orthogonal sequence same as in the transmitter and the 8 bit output from serial to parallel converter. The output despreaded bit from the Despreading block is then collected by serial to parallel converter 2 to form a byte data. This byte of data is then displayed in the display device (LCD and seven segments).

The components used in receiver can be listed as follows-

- Serial to parallel converter
- Despreading Block
- LCD interfacing module

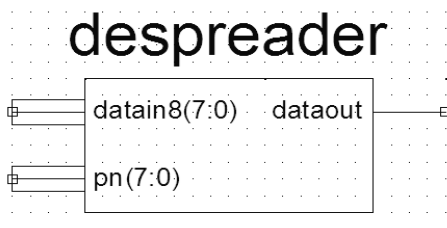
Serial to parallel converter

Two serial to parallel converters are used, the first one is used to receive the transmitted serial data and the other to combine individual bits of an 8 bit data.



Despreading Block

The Despreading module accepts 8 bit spreaded data and corresponding orthogonal sequence for each user and outputs the corresponding despreaded bit.



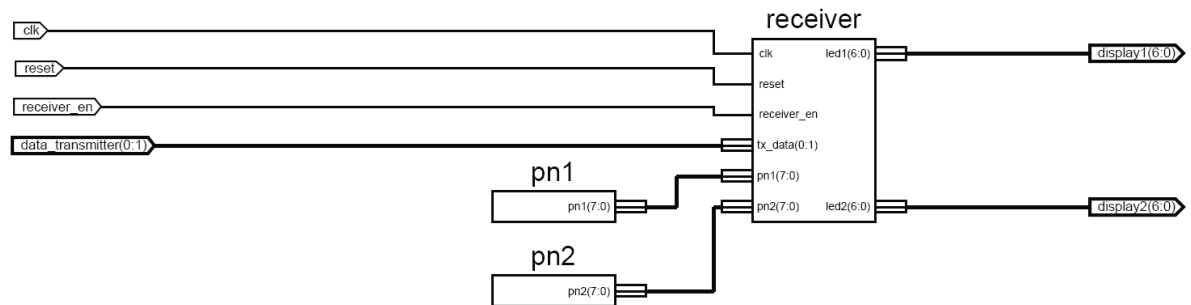
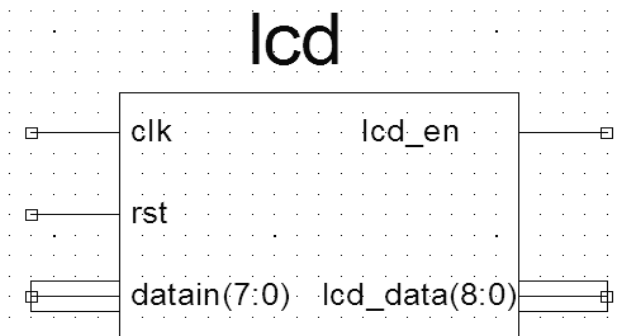


Figure 3.2: Receiver top level module

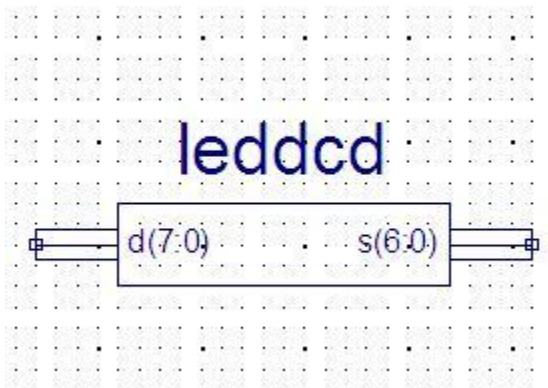
LCD interfacing module

The LCD interfacing module accepts the 8 bit ASCII user data and outputs the corresponding data and signals en(lcd_data(8)) and RS for the LCD.



7-segment decoder module

This module stores the look up table for displaying values '0' to '9' in the 7-segment display.



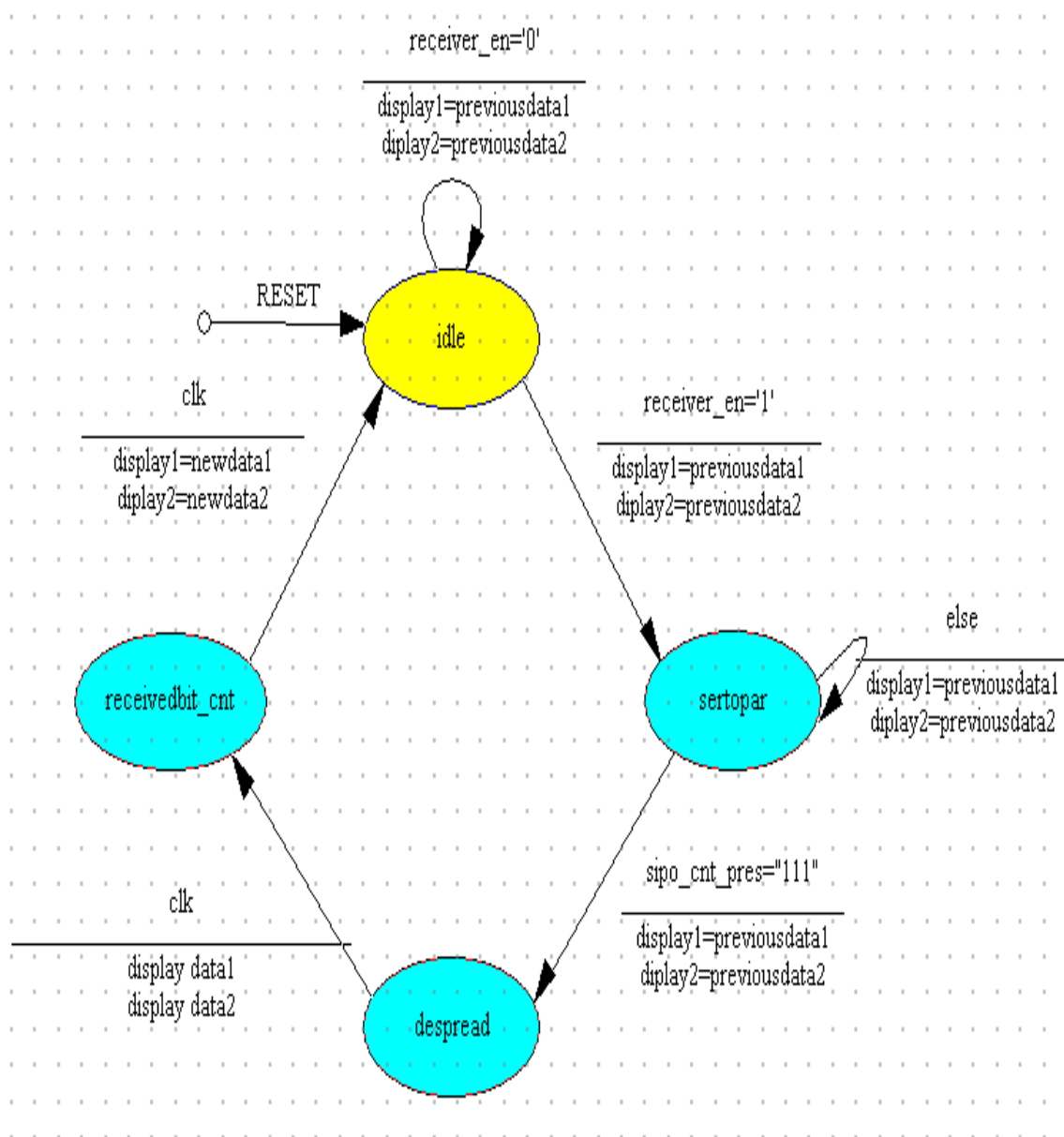


Figure 3.3: Receiver State Diagram

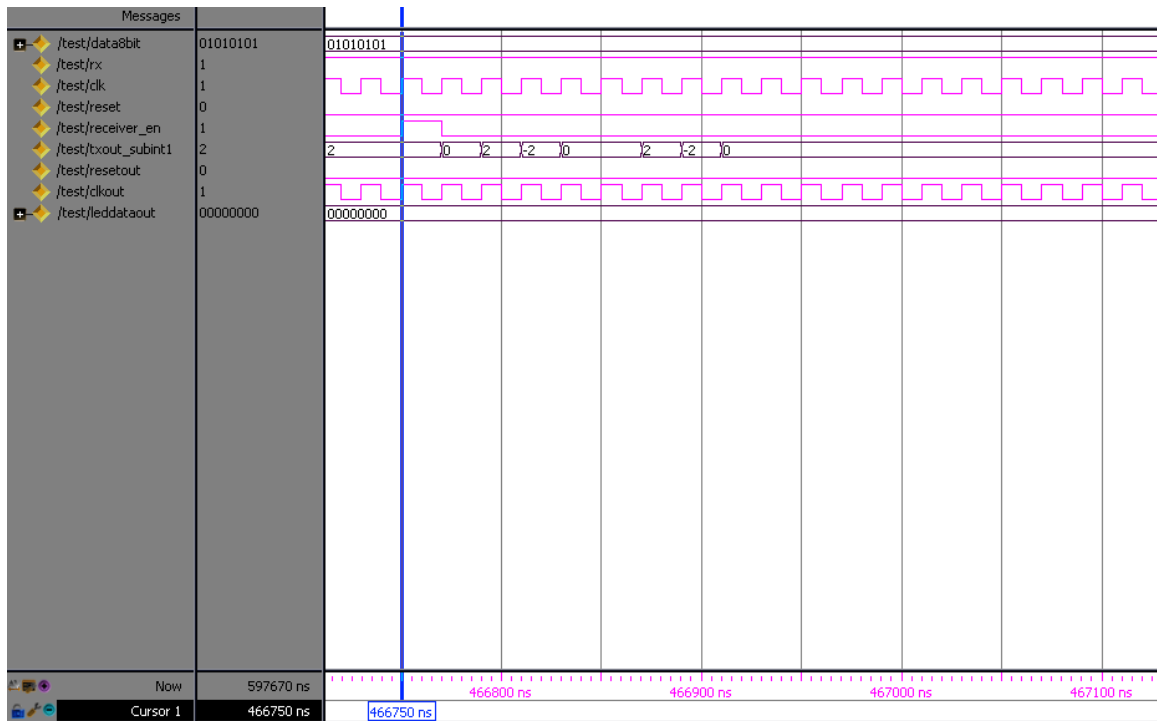


Figure 3.4: Simulation Results of Transmitting FPGA

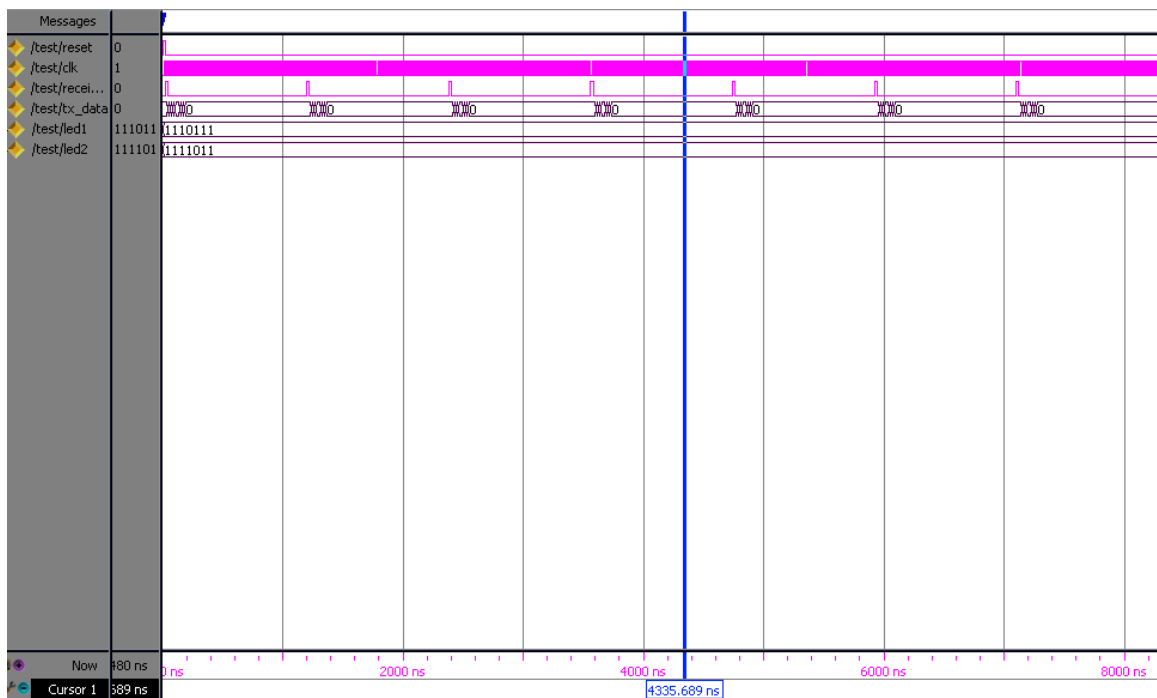
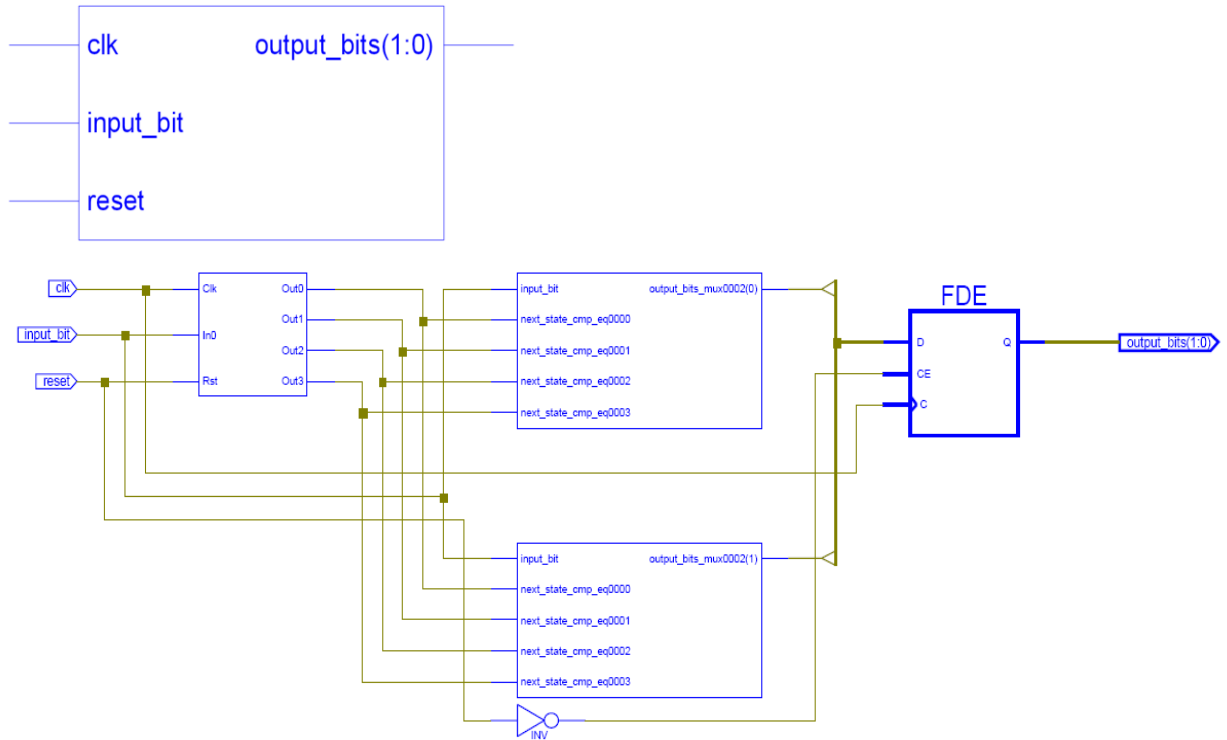


Figure 3.5: Simulation Results of Receiving FPGA

3.2.3 Convolutional Encoder

The convolutional encoder used is of rate $\frac{1}{2}$ and constraint length 3. The generator matrix are $g_1=(1\ 1\ 1)$ and $g_2=(1\ 0\ 1)$.

The encoder produces 2 bit output for each input bit.



]

Figure 3.6: RTL schematic of convolutional encoder

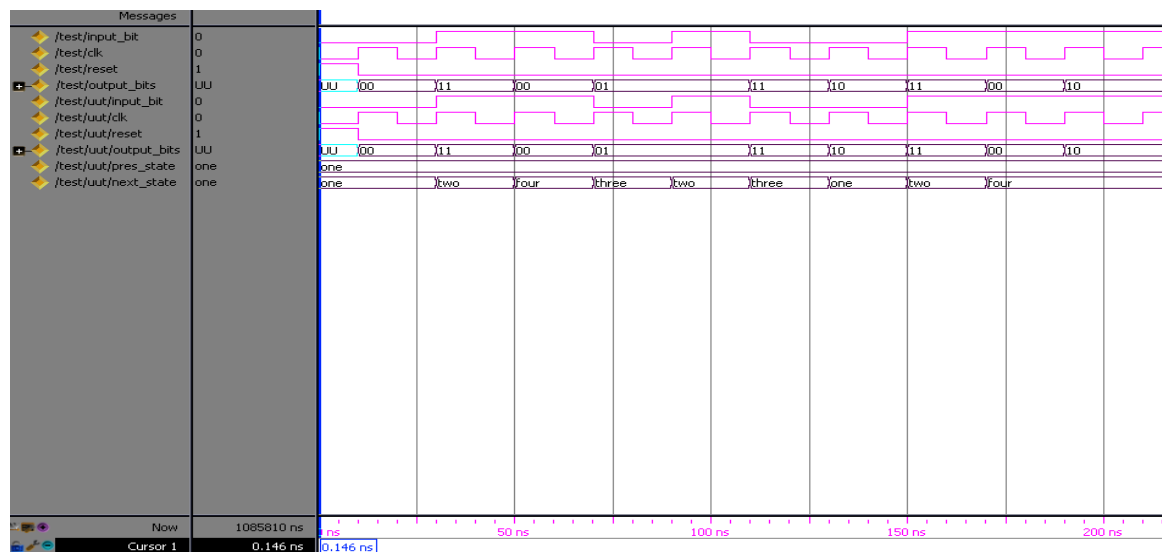


Figure 3.7: Simulation Results of Convolutional Encoder

3.3 Interfacing Computer through Parallel and Serial Ports:

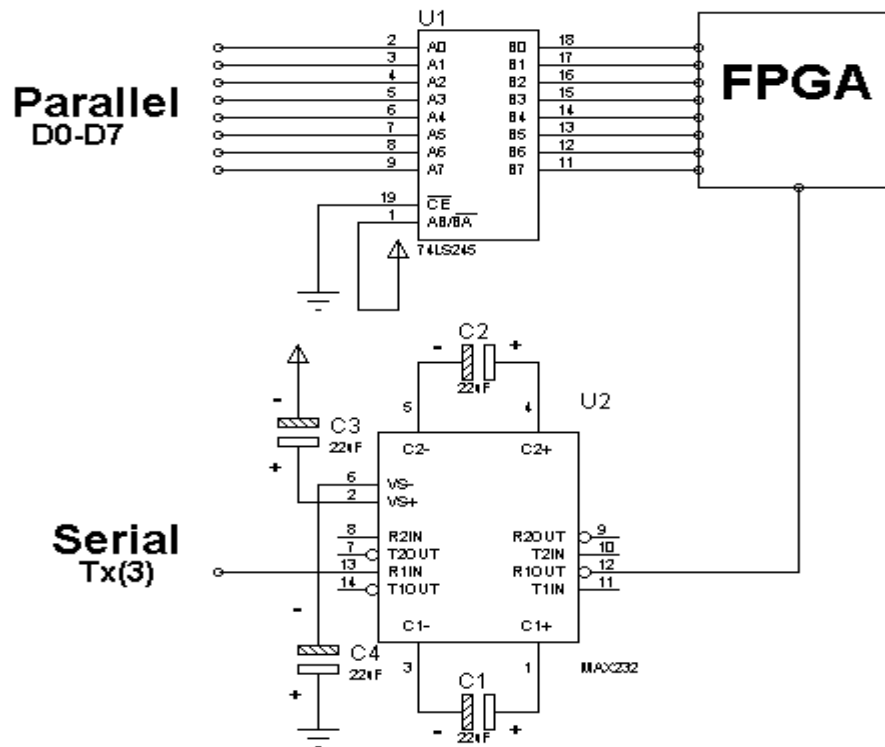


Figure 3.8: Interfacing FPGA with PC's Parallel and Serial ports

Buffer 74LS245 is used as an interfacing unit between parallel port and the FPGA input pins so as to protect the parallel port from sourcing large currents which might damage the parallel port and I/O pins of the FPGA. In this case 8-bit data available to the buffer through the parallel port is buffered into output lines which are received at the input pins of the FPGA.

Parallel Port Pins	Port	Buffer In	Buffer Out	FPGA In
2 (D0)		2	18	P86
3		3	17	P87
4		4	16	P94
5		5	15	P12
6		6	14	P13
7		7	13	P19
8		8	12	P20
9 (D7)		9	11	P21
25 (GND)		10 (GND)	10 (GND)	(GND)

Table 3.1: Interfacing PC and FPGA

Max232 is used as an interfacing unit between serial port and the FPGA input pin. The chip converts the RS232 logic levels into corresponding TTL and vice-versa. Since the serial data is transmitted from PC to FPGA, Tx (pin 3) of the serial port is connected to the R1IN (pin 13) of Max232 which converts this RS232 logic signals into corresponding TTL logic into R1OUT (pin 12). The TTL output is fed to transmitting FPGA at pin p78. The serial data and the FPGA both share the common GND.

3.4 Interfacing 7 Segment Display with Line Driver 74LS245:

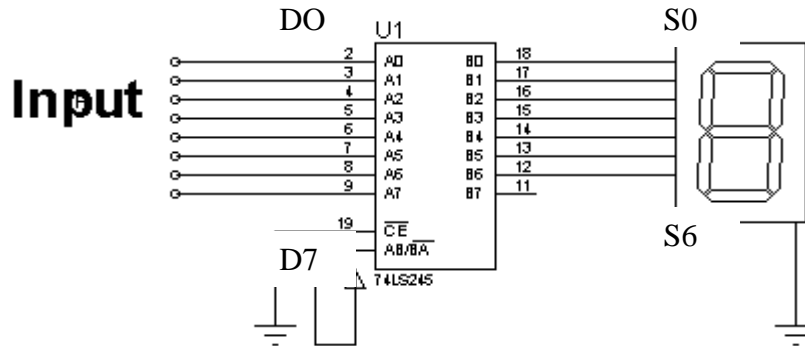


Figure 3.9: Interfacing 7 Segment Display with Line Driver 74LS245

The line driver 74LS245 (buffer) is used to drive the output 7-segment display so that the input lines are not loaded. In the project, the buffer is utilized

To interface an extra 7 segment display at the receiver so that two user data can be displayed into two 7segment displays. In this case there are only 7 pins out from FPGA which is provided to the buffer and the extra 1 pin left for input is grounded to balance the circuit.

3.5 Final Circuit Diagram:

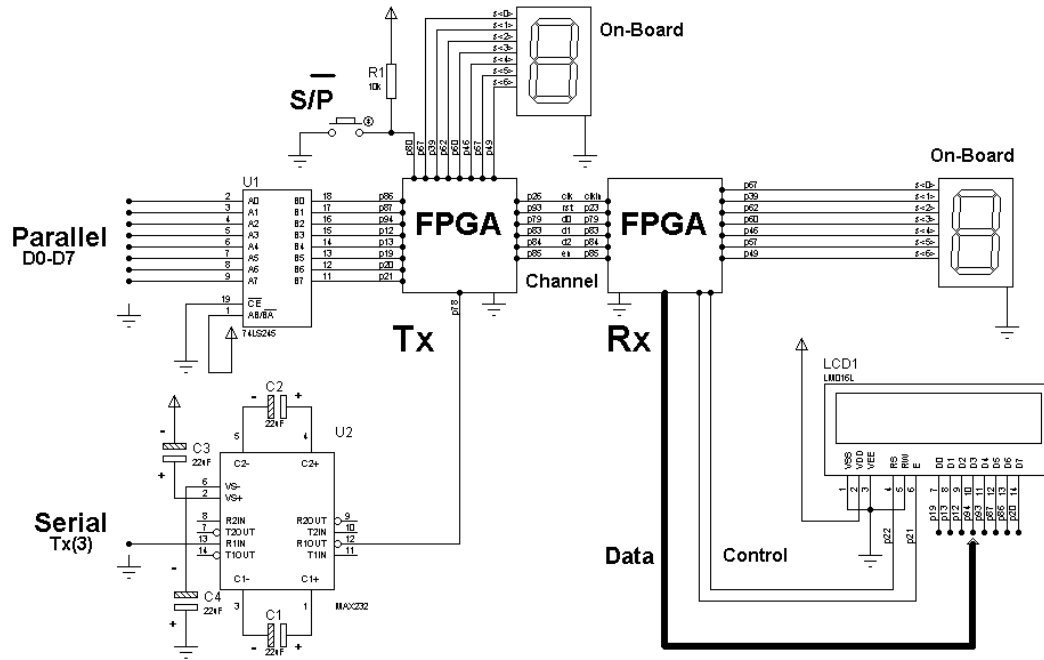


Figure 3.10: Final Phase Circuit Diagram

- Parallel and Serial User Data are provided through PC to the transmitting FPGA.
- The toggle switch S/P is used to display the value of the serial and the parallel data at the transmitting side.
- Two FPGAs are connected via three data lines, receiver_en.
- Common clock and reset are provided to both FPGAs.
- The serial data and parallel data at the receiver FPGA is displayed in the 7-segment display and LCD respectively.

Chapter 4: Software Simulation (MATLAB)

Matlab Simulations:

Matlab GUI was used as a tool to simulate the principles of CDMA. In the simulation sliders are utilized, edit texts and axes of the MATLAB GUI.

The corresponding GUI of MATLAB is invoked through 'GUIDE' function. The following GUI objects are used in the project.

Sliders:

The sliders are used to provide user data between numbers 0 and 9. The position of the slider changes the value that is provided to both the simulation process and FPGA hardware.

```
set(handles.slider0,'Value',bi2de(handles.data0,'left-msb'))
```

Sets the position of the slider according to the 8-bit binary value of 'data0'.

```
slider_value0=fix(get(handles.slider0,'Value'))
```

Returns the value of the slider position to a variable 'slider_value0'.

Edit texts:

The edit texts are utilized to display which data is sent through the sliders.

```
set(handles.edit2,'String',num2str(slider_value0))
```

Displays the value of the 'slider_value0' in edit text 'edit2'.

Axes:

The axes is used to plot the value of the variables for proper visualization.

```
axes(handles.data0_axes);
```

```
stairs([data0 0]);
```

```
axis([1 9 -.1 1.1]);
```

```
set(handles.data0_axes,'XMinorTick','On');
```

```
grid on;
```

This plots the value of 'data0' into axis 'data0_axes' in stairs with X and Y axis limits.

Apart from GUI the following functions have been constructed:

Encode8: Spreads 8-bit data using 8-bit code into 64-bit result.

Decode8: De-spreads 64-bit data using 8-bit code into 8-bit data.

Polar8: Converts the binary 8-bit data (1's and 0's) into polar form (1 and -1).

My_conv_enc: Spreaded 64-bit data are encoded through a convolutional encoder (Rate 1/3) resulting in a 64*3 bit data.

My_conv_dec: The convolutional encoder data 64*3 bit are decoder through a convolutional decoder (Rate 3/1) resulting in a 64 bit data which is later de-spreaded.

Simple_slider.fig:

This is the file where the GUI is created for the project and define the ‘callback’ function for each tool.

Simple_slider.m: This is the main file that is to be run.

(Classes like ‘digitalio’ and ‘serial’ are utilized to define provide data parallelly and serially to real hardware).

4.1 Analog Simulation:

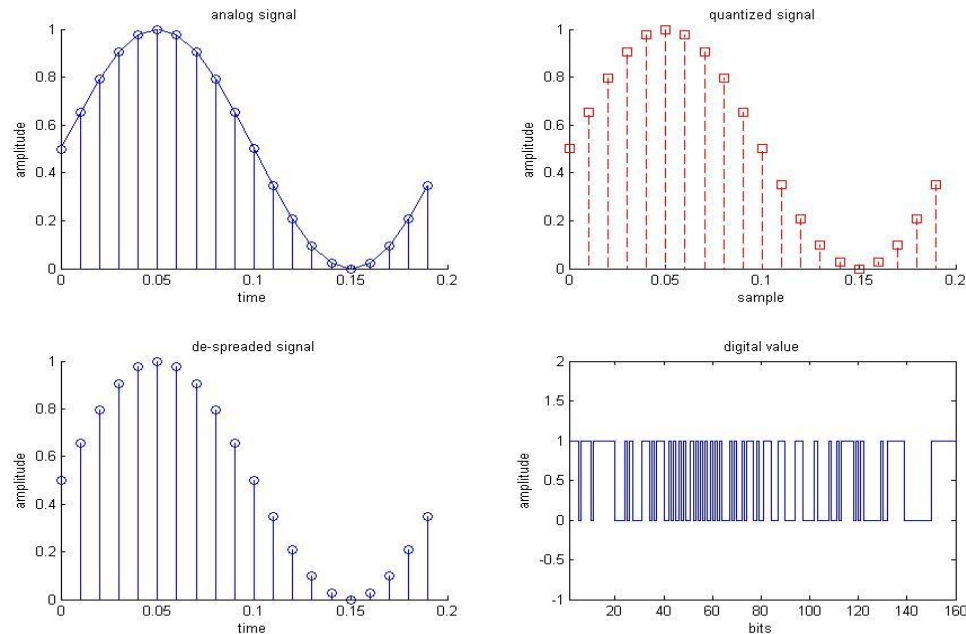


Figure 4.1: Analog Simulation

The figure demonstrates an analog simulation of CDMA principles for a single user. An input signal is digitized and the digitized value is spreaded and de-spreaded to obtain the original signal.

- Fig 1 shows sampled values of a pure sinusoidal signal.
- Fig 2 (red) shows its quantized value obtain through ‘quantiz’ function of the MATLAB.
- Fig 4 shows the corresponding 8-bit digital values of each quantized result with ‘1’ as the reference value.
- The digital value is spreaded using 8-bit orthogonal code in transmitter, which is de-spreaded by the same 8 bit orthogonal code in the receiver. Fig 3 shows the recovery of the original signal.

(The details of the spreading and de-spreading of digital values will be explained in the digital simulation.)

4.2 Digital Simulation:

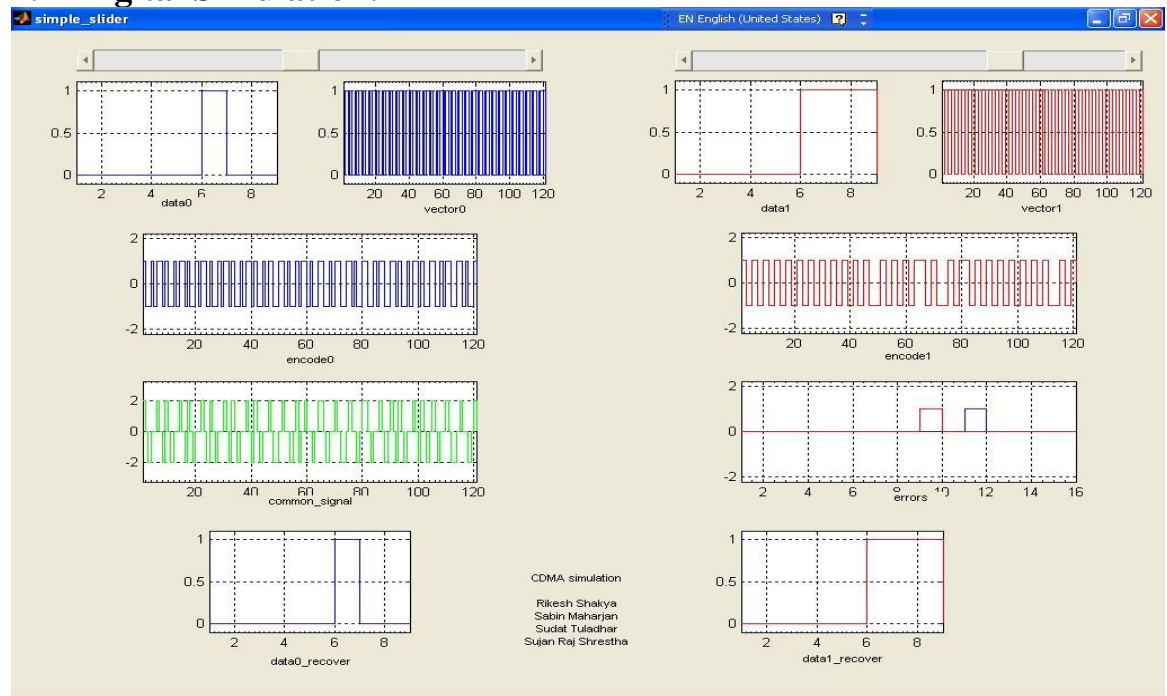


Figure 4.2: Digital Simulation (with channel coding)

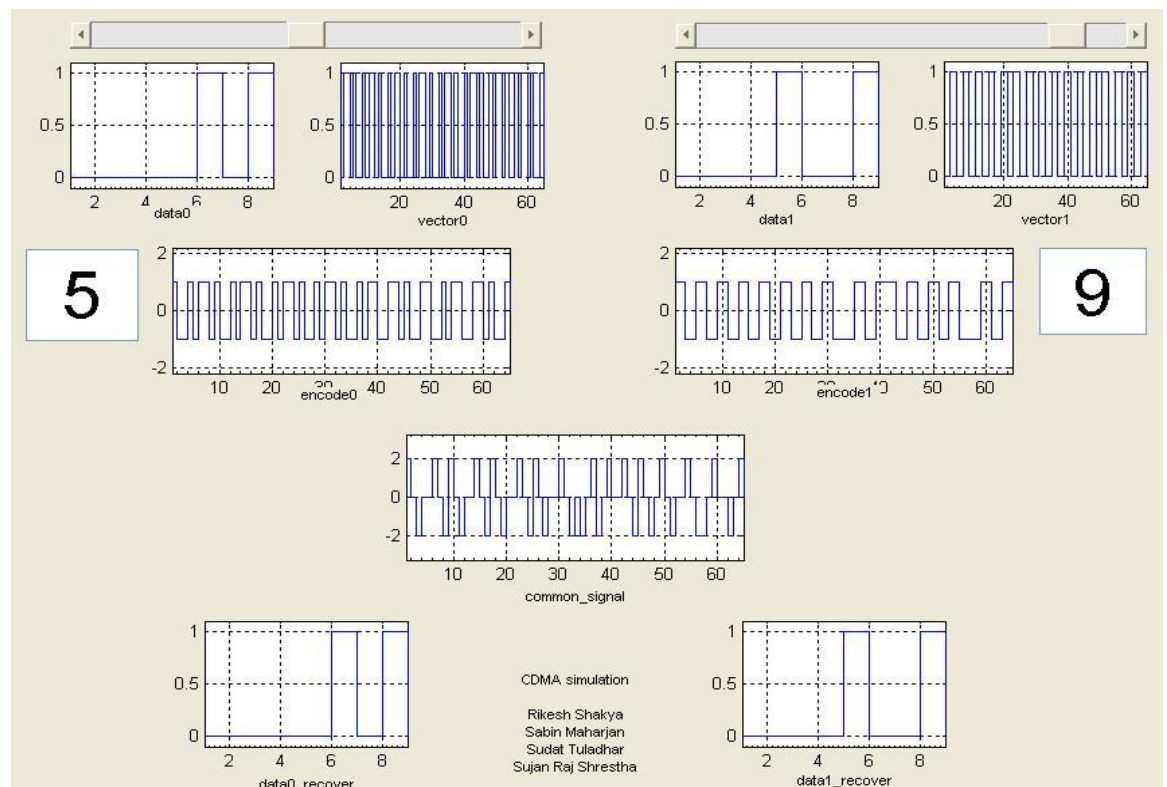


Figure 4.3: Digital Simulation (without channel coding)

The figures 1 and 2 demonstrate digital simulation of CDMA principles for two users. Two scroll bars can be used to vary two digital input values ('0' to '9'), which would be fed to the FPGA via parallel and the serial port respectively. (Inputs are limited to characters '0' to '9' since our display device is the 7-segment display). Same inputs are used to drive the simulation also. Simulation uses value 0 for '0' character while sending through parallel and serial ports 48 is sent for '0'.

In **figure 4.2:** (Including channel coding)

- Each data (data0 and data1) is spreaded using two 8-bit unique orthogonal codes denoted by vector0 ('blue') and vector1 ('red'), which results in spreaded data encode0 and encode1.
- Both spreaded data encode0 and encode1 are merged into a common_signal ('green') in a channel.
- In order to verify the channel coding scheme, errors are introduced into the channel in a random manner utilizing randerr () function. Each error bit (blue and red) in error figure denotes the error that is introduced into the channel deliberately.
- Despite errors, the data is recovered into its original form.

In **figure 4.3:** (Excluding channel coding)

- This is the actual GUI used to demonstrate the spreading and despreading process through simulation and hardware.
- Everything else is same but here the textbox are added to show which data is actually being transmitted through MATLAB for both users.

Chapter 5: Cost Estimation

S.No	Components	Nos.	Cost per unit(Rs)	Cost(Rs)
1.	Max 232	1	125	125
2.	Capacitors (22 uF)	4	10	40
3.	Buffer (74LS245)	2	60	120
4.	7-Segment Display (CC)	1	35	35
5.	LCD	1	650	650
6.	Serial Cable	1	60	60
7.	Parallel Cable	1	60	60
8.	LEDs			20
9.	Registers			40
10.	Push Buttons			10
11.	Hook-Up wires			40
12.	FPGA	2	6,400 (\$80)	12,800(\$160)
	Total Cost			Rs 14,000

Table 5.1: Cost Estimate of the project

Labour cost: Rs 500 per hour.

Total Working hours: 45 working days (6 working hours a day)

Total Labour Cost: Rs 1, 35,000

(The project group consists of four individuals and the cost estimates above reflect labour cost of the overall group)

Chapter 6: Epilogue

5.1 Limitations

- The system model is focused on the spreading, de-spreading and the channel coding the user data.
- User input data is purely digital.
- The system model is only unidirectional.
- The wired channel is used. Hence limited to short distances.
- Convolutional Coding is limited to simulations only.
- FPGA to FPGA communication is done through multi-wire.
- Clock recovery is not done in the receiver; same clocks are explicitly provided to both transmitters and receivers.

5.2 Future Enhancement

- The complete CDMA system including voice-coding scrambling, interleaving, modulation and wireless transmission can be integrated.
- The system can be optimized for voice transmission.
- The system can also be optimized for bi-directional data transfer.
- FPGA to FPGA communication via UART could be accomplished which would require only a single wire, which is practically implementable.
- Clock recovery circuitry could be built into receivers.

5.3 Difficulties faced in the project

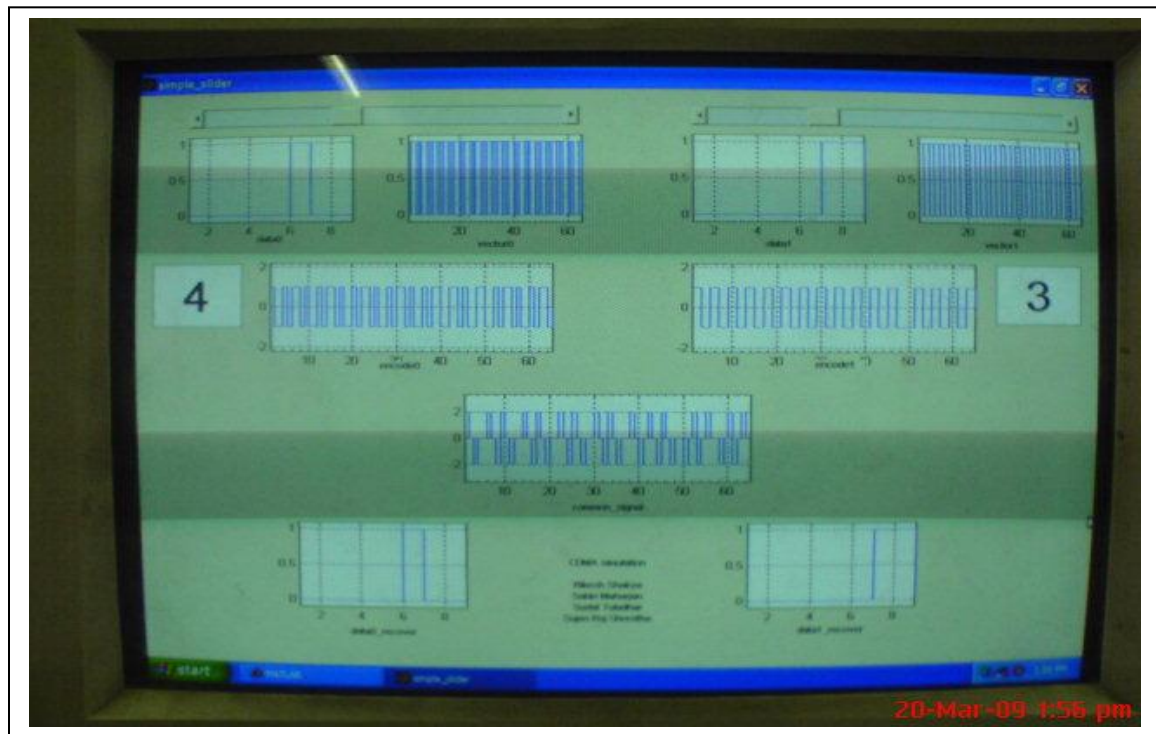
- VHDL involves concurrent programming, since most of traditional programming languages are sequential it took time to adjust with this concurrent operation environment.
- VHDL consists of many structures which can only be simulated, during project implementation these structures often troubled during the implementation of design.
- The Xilinx ISE 10.1 tool used in the project is found to contain several bugs this often troubled us during the design implementation process.
- Synchronization of various modules in FPGA was difficult.
- Implementation of channel coding was difficult and time consuming.

5.4 Conclusion

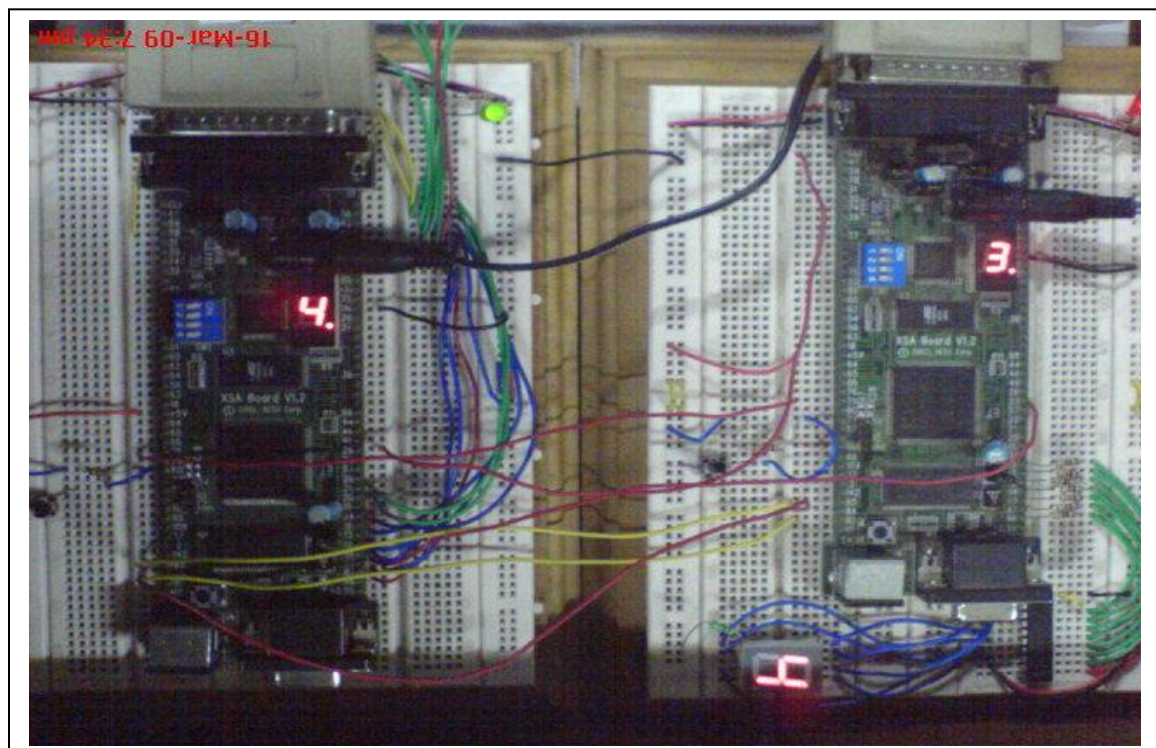
A complete Spread Spectrum communication link requires various advanced and up-to-date technologies and disciplines: RF antenna, powerful and efficient PA, low-noise, highly linear LNA, compact transceivers, high-resolution ADC's and DACs, rapid low-power digital signal processing (DSP), etc.

But the effort in hardware synthesis of CDMA can be very useful projects at bachelor level which can help the students learn the real applications of advanced digital communication and HDL programming. With a lot of thinking and references, it can be concluded that the hardware realization of CDMA Technique is feasible despite of the limited resources and expertise.

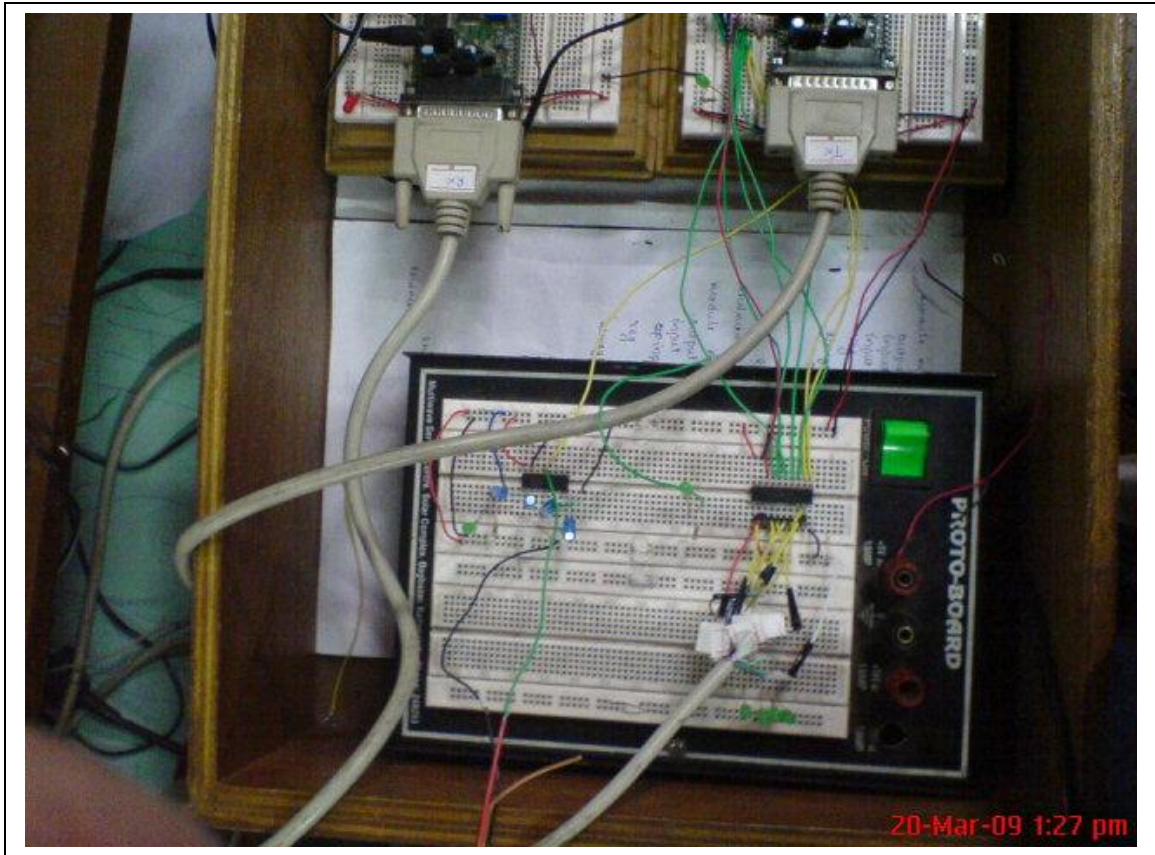
System Operation Snapshots:



MATLAB Simulation



Hardware Verification



Interfacing PC with FPGA via MAX232 Voltage Converter

Appendix



XSA Board V1.1, V1.2 User Manual

How to install, test, and use
your new XSA Board

Copyright © 2001-2004 by X Engineering Software Systems Corporation.

All XS-prefix product designations are trademarks of XESS Corp.

All XC-prefix product designations are trademarks of Xilinx.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.




Table of Contents

Table of Contents	2
Preliminaries	4
Getting Help!	4
Take notice!!	4
Packing List	5
Installation	6
Installing the XSTOOLS Utilities and Documentation	6
Applying Power to Your XSA Board	6
Using a 9VDC wall-mount power supply	6
Powering Through the PS/2 Connector	6
Solderless Protoboard Installation	6
Connecting a PC to Your XSA Board	8
Connecting a VGA Monitor to Your XSA Board	8
Connecting a Mouse or Keyboard to Your XSA Board	9
Inserting the XSA Board into an XStend Board	9
Setting the Jumpers on Your XSA Board	9
Testing Your XSA Board	10
Setting the XSA Board Clock Oscillator Frequency	11
Programming	13
Downloading Designs into the FPGA and CPLD of Your XSA Board	13
Storing Non-Volatile Designs in Your XSA Board	16
Downloading and Uploading Data to the SDRAM in Your XSA Board...	18
Programmer's Models	20
XSA Board Organization	20
Programmable logic: Spartan-II FPGA and XC9572XL CPLD	21

100 MHz Programmable Oscillator	21
Synchronous DRAM	23
Flash RAM	24
Seven-Segment LED	25
Four-Position DIP Switch	25
PS/2 Port.....	25
Pushbutton.....	26
VGA Monitor Interface	26
Parallel Port Interface	26
Prototyping Header	29
XSA Pin Connections	31
XSA Schematics	32

1

Preliminaries

Getting Help!

Here are some places to get help if you encounter problems:

- If you can't get the XSA Board hardware to work, send an e-mail message describing your problem to help@xess.com or submit a problem report at <http://www.xess.com/help.html>. Our web site also has
 - [answers to frequently-asked-questions](#),
 - [example designs, application notes and tutorials for the XS Boards](#),
 - [a place to sign-up for our email forum](#) where you can post questions to other XS Board users.
- If you can't get your Xilinx WebPACK software tools installed properly, send an e-mail message describing your problem to hotline@xilinx.com or check their web site at <http://www.xilinx.com/support/support.htm>.
- If you need help using the WebPACK software to create designs for your XSA Board, then check out this [tutorial](#).

Take notice!!

- The XSA Board requires an external power supply to operate! It does not draw power through the downloading cable from the PC parallel port.
- If you are connecting a 9VDC power supply to your XSA Board, please make sure the center terminal of the plug is positive and the outer sleeve is negative.
- Do not power your XSA Board with a battery! This will not provide enough current to insure reliable operation of the XSA Board.

Packing List

Here is what you should have received in your package:

- an XSA Board;
- a 6' cable with a 25-pin male connector on each end;
- an XSTOOLS CDROM with software utilities and documentation for using the XSA Board.

2

Installation

Installing the XSTOOLS Utilities and Documentation

Xilinx currently provides the WebPACK tools for programming their CPLDs and Spartan-II FPGAs. The XESS CDROM contains a version of WebPACK that will generate bitstream configuration files compatible with your XSA Board. You can also [download](#) the most current version of the WebPACK tools from the Xilinx website..

In addition, XESS Corp. provides the XSTOOLS utilities for interfacing a PC to your XSA Board. Run the SETUP.EXE program on the XSTOOLS CDROM to install these utilities.

Applying Power to Your XSA Board

You can use your XSA Board in three ways, distinguished by the method you use to apply power to the board.

Using a 9VDC wall-mount power supply

You can use your XSA Board all by itself to experiment with logic designs. Just place the XSA Board on a non-conducting surface as shown in Figure 1. Then apply power to jack J5 of the XSA Board from a 9V DC wall-mount power supply with a 2.1 mm female, center-positive plug. (See Figure 2 for the location of jack J5 on your XSA Board.) The on-board voltage regulation circuitry will create the voltages required by the rest of the XSA Board circuitry. **Be careful!! The voltage regulators on the XSA Board will become hot.** Attach a heat sink to them if necessary.

Powering Through the PS/2 Connector

You can use your XSA Board with a laptop PC by connecting a PS/2 male-to-male cable from the PS/2 port of the laptop to the J4 connector. You must also have a shunt across pins 1 and 2 of jumper J7. The on-board voltage regulation circuitry will create the voltages required by the rest of the XSA Board circuitry. **Many PS/2 ports cannot supply more than 0.5A so large, fast FPGA designs may not work when using this power source!**

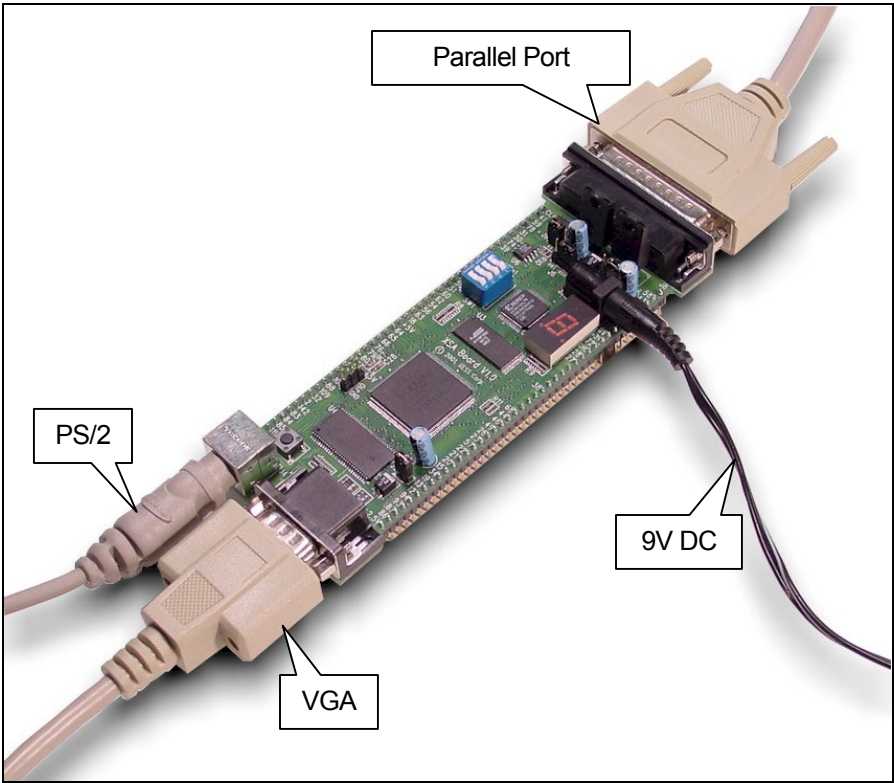
Solderless Protoboard Installation

The two rows of pins from your XSA Board can be plugged into a solderless protoboard with holes spaced at 0.1" intervals. (One of the A.C.E. protoboards from 3M is a good choice.) Once plugged in, many of the pins of the FPGA are accessible to other circuits on the protoboard. (The numbers printed next to the rows of pins on your XSA Board

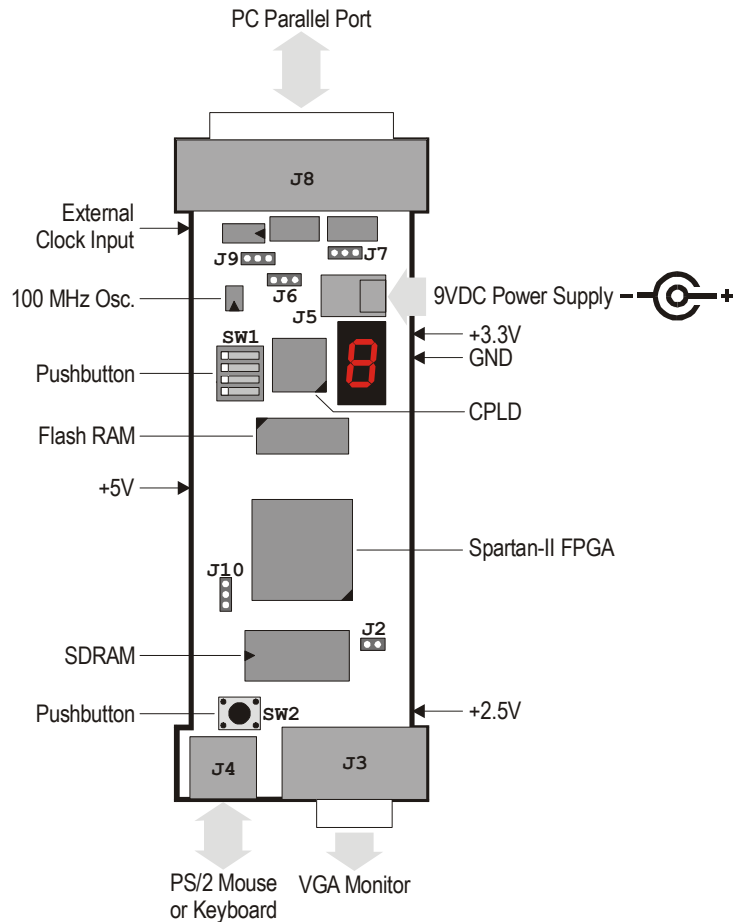
correspond to the pin numbers of the FPGA.) Power can still be supplied to your XSA Board though jack J5, or power can be applied directly through several pins on the underside of the board. Just connect +5V, +3.3V, +2.5V and ground to the pins of your XSA Board listed in Table 1.

• Table 1: Power supply pins for the XSA Board.

Voltage	Pin	Note
+5V	2	
+3.3V	22	Remove the shunt from jumper J7 if you wish to use your own +3.3V supply. Leave the shunt on jumper J7 to generate the +3.3V supply from the +5V supply.
+2.5V	54	Remove the shunt from jumper J2 if you wish to use your own +2.5V supply. Leave the shunt on jumper J2 to generate the +2.5V supply from the +3.3V supply.
GND	52	



• Figure 1: External connections to the XSA Board.



• Figure 2: Arrangement of components on the XSA Board.

Connecting a PC to Your XSA Board

The 6' DB25 male-to-male cable included with your XSA Board connects it to a PC. One end of the cable attaches to the parallel port on the PC and the other connects to the female DB-25 connector (J8) at the top of the XSA Board as shown in Figure 1.

Connecting a VGA Monitor to Your XSA Board

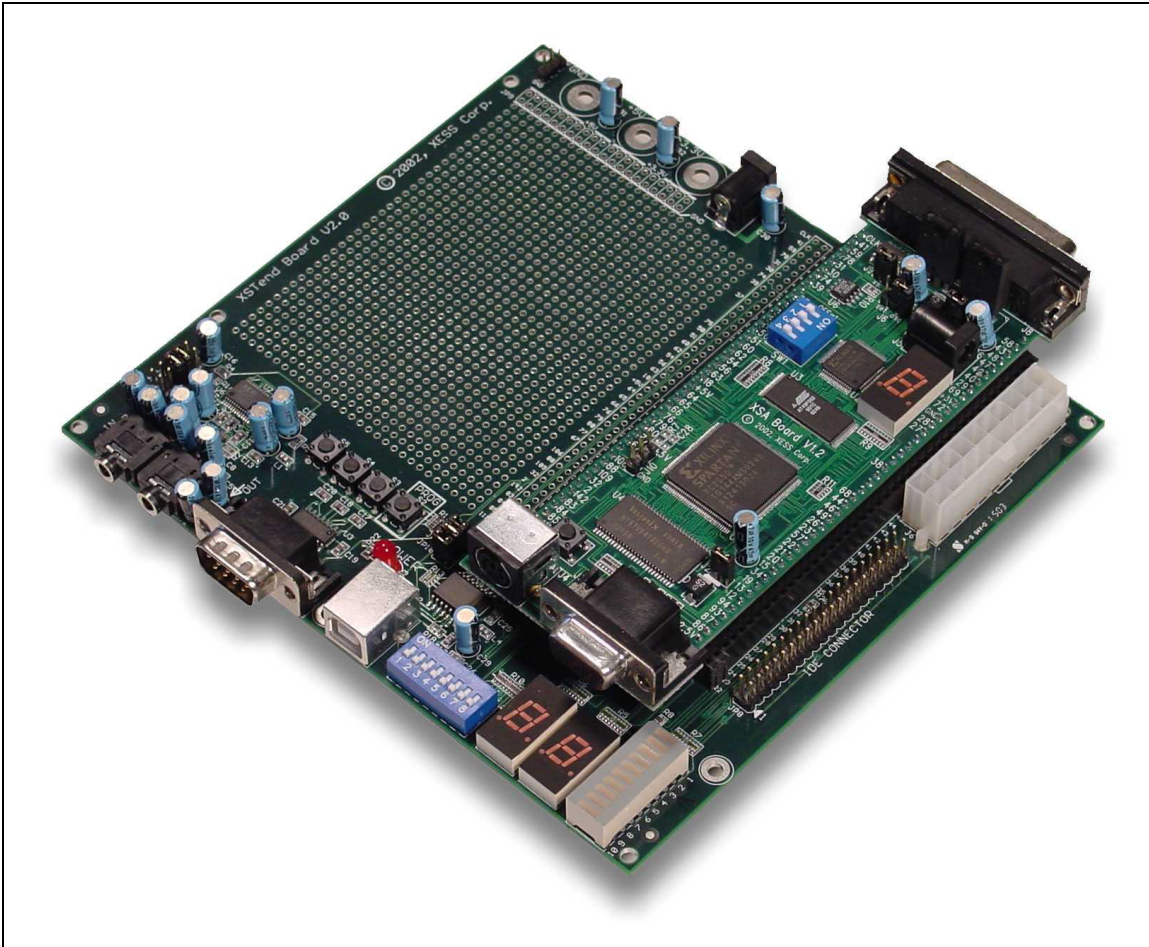
You can display images on a VGA monitor by connecting it to the 15-pin J3 connector at the bottom of your XSA Board (see Figure 1). You will have to create a VGA driver circuit for your XSA Board to actually display an image. You can find an example VGA driver at <http://www.xess.com/ho03000.html>.

Connecting a Mouse or Keyboard to Your XSA Board

You can accept inputs from a keyboard or mouse by connecting it to the J4 PS/2 connector at the bottom of your XSA Board (see Figure 1). You can find an example keyboard driver at <http://www.xess.com/ho03000.html>.

Inserting the XSA Board into an XStend Board

If you purchased the optional XST-2.x Board, then the XSA Board is inserted as shown below. Refer to the XST-2.x Board Manual for more details.



Setting the Jumpers on Your XSA Board

The default jumper settings shown in Table 2 configure your XSA Board for use in a logic design environment. You will need to change the jumper settings only if you are:

- downloading FPGA bitstreams to your XSA Board using the Xilinx iMPACT software;

- reprogramming the clock frequency on your XSA Board (see page 11);
- changing the power sources for the XSA supply voltages.

• Table 2: Jumper settings for XSA Boards.

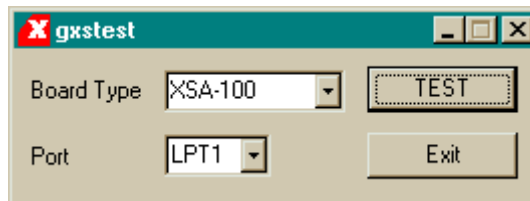
Jumper	Setting	Purpose
J2	On (default)	A shunt should be installed if the +2.5V supply voltage is derived from the +3.3V supply.
	Off	The shunt should be removed if the +2.5V supply voltage is applied from an external source through pin 22 of the XSA Board (labeled "+2.5V" at the lower right-hand corner of the board).
J6	1-2 (set)	The shunt should be installed on pins 1 and 2 (set) when setting the frequency of the programmable oscillator.
	2-3 (osc) (default)	The shunt should be installed on pins 2 and 3 (osc) during normal operations when the programmable oscillator is generating a clock signal.
J7	1-2 (default)	The shunt should be installed on pins 1 and 2 if the +3.3V supply voltage is derived from the +5V supply.
	2-3	The shunt should be installed on pins 2 and 3 if the +3.3V supply voltage is derived from the 9VDC supply applied through jack J5.
J9	1-2 (xi)	The shunt should be installed on pins 1 and 2 (xi) if the XSA Board is to be downloaded using the Xilinx iMPACT software.
	2-3 (xs) (default)	The shunt should be installed on pins 2 and 3 (xs) if the XSA Board is to be downloaded using the XESS GXSLD software.
J10	N/A	This is a header that provides access to the +5V and GND references on the board. No shunt should be placed on this header.

Testing Your XSA Board

Once your XSA Board is installed and the jumpers are in their default configuration, you can test the board using the GUI-based GXSTEST utility as follows.



You start GXSTEST by clicking on the GXSTEST icon placed on the desktop during the XSTOOLS installation. This brings up the window shown below.



Next you select the parallel port that your XSA Board is connected to from the Port pulldown list. GXSTEST starts with parallel port LPT1 as the default, but you can also select LPT2 or LPT3 depending upon the configuration of your PC.

After selecting the parallel port, you select either the XSA-50 or XSA-100 item in the Board Type pulldown list. Then click on the TEST button to start the testing procedure. GXSTEST will configure the FPGA to perform a test procedure on your XSA Board.

Within thirty seconds you will see a O displayed on the LED digit if the test completes successfully. Otherwise an E will be displayed if the test fails. A status window will also appear on your PC screen informing you of the success or failure of the test.

If your XSA Board fails the test, you will be shown a checklist of common causes for failure. If none of these causes applies to your situation, then test the XSA Board using another PC. In our experience, 99.9% of all problems are due to the parallel port. If you cannot get your board to pass the test even after taking these steps, then contact XESS Corp for further assistance.


As a result of testing the XSA Board, the CPLD is programmed with the standard parallel port interface found in the dwnldpar.svf bitstream file located within the XSTOOLS\XSA folder. This is the standard interface that should be loaded into the CPLD when you want to use it with the GXSLoad utility.

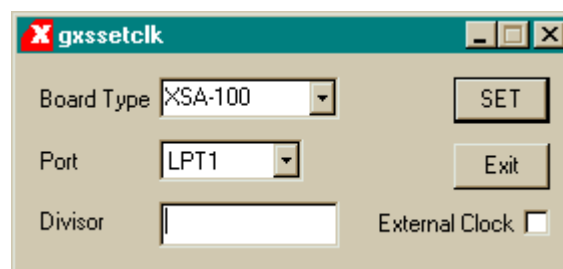
Setting the XSA Board Clock Oscillator Frequency

The XSA Board has a 100 MHz programmable oscillator (a Dallas Semiconductor DS1075Z-100). The 100 MHz master frequency can be divided by factors of 1, 2, ... up to 2052 to get clock frequencies of 100 MHz, 50 MHz, ... down to 48.7 KHz, respectively. The divided frequency is sent to the rest of the XSA Board circuitry as a clock signal.

The divisor is stored in non-volatile memory in the oscillator chip so it will resume operation at its programmed frequency whenever power is applied to the XSA Board. You can store a particular divisor into the oscillator chip by using the GUI-based GXSETCLK as follows.



You start GXSETCLK by clicking on the  icon placed on the desktop during the XSTOOLS installation. This brings up the window shown below.



Your next step is to select the parallel port that your XSA Board is connected to from the Port pulldown list. Then select either XSA-50 or XSA-100 in the Board Type pulldown list.

Next you enter a divisor between 1 and 2052 into the Divisor text box and then click on the SET button. Then follow the sequence of instructions given by XSSETCLK for moving shunts and removing and restoring power during the oscillator programming process. At the completion of the process, the new frequency will be programmed into the DS1075.

An external clock signal can be substituted for the internal 100 MHz oscillator of the DS1075. Checking the External Clock checkbox will enable this feature in the

programmable oscillator chip. If this option is selected, you are then responsible for providing the external clock to the XSA Board through pin 64 (labeled “CLK” at the upper left-hand corner of the board).

3

Programming

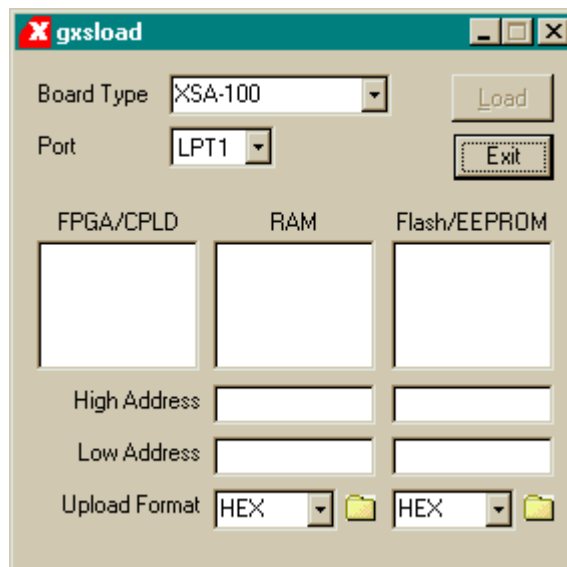
This section will show you how to download a logic designs into the FPGA and CPLD of your XSA Board and how to download and upload data to and from the SDRAM and Flash devices on the board.

Downloading Designs into the FPGA and CPLD of Your XSA Board

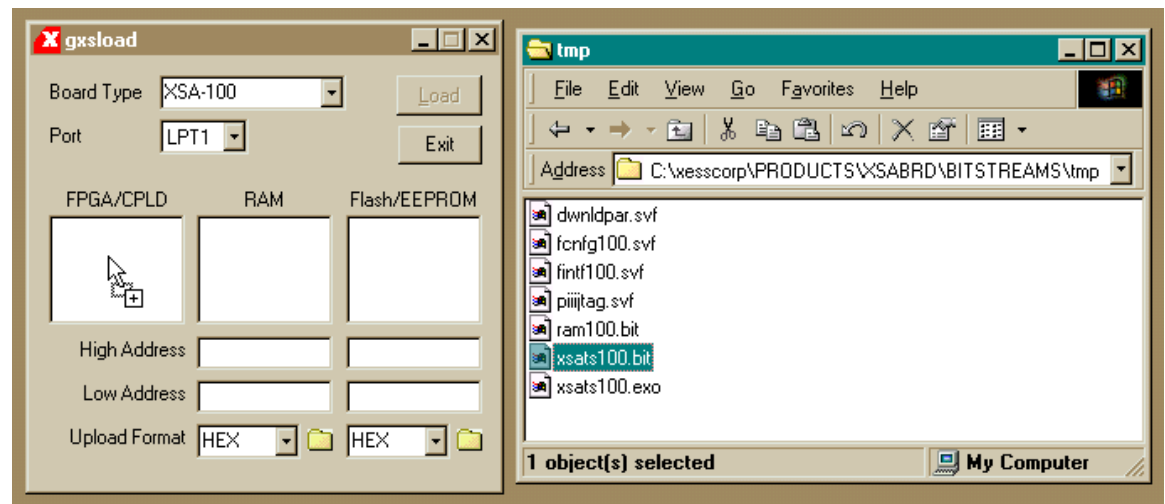
During the development and testing phases, you will usually connect the XSA Board to the parallel port of a PC and download your circuit each time you make changes to it. You can download a Spartan-II FPGA design into your XSA Board using the GXSLD utility as follows.



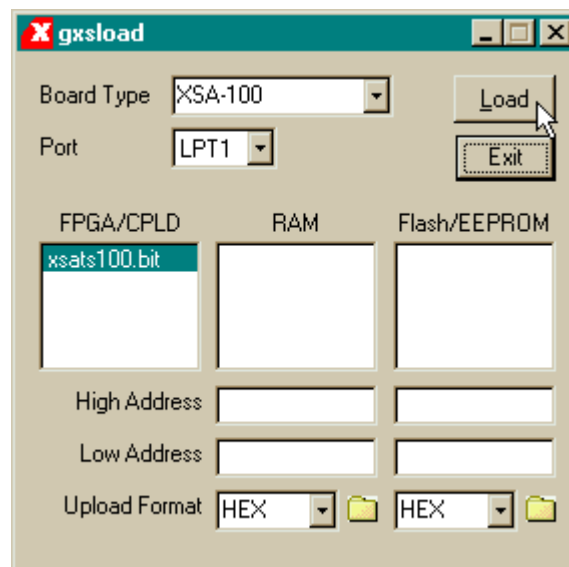
You start GXSLD by clicking on the GXSLD icon placed on the desktop during the XSTOOLS installation. This brings up the window shown below. Then select the type of XS Board you are using and the parallel port to which it is connected as follows.



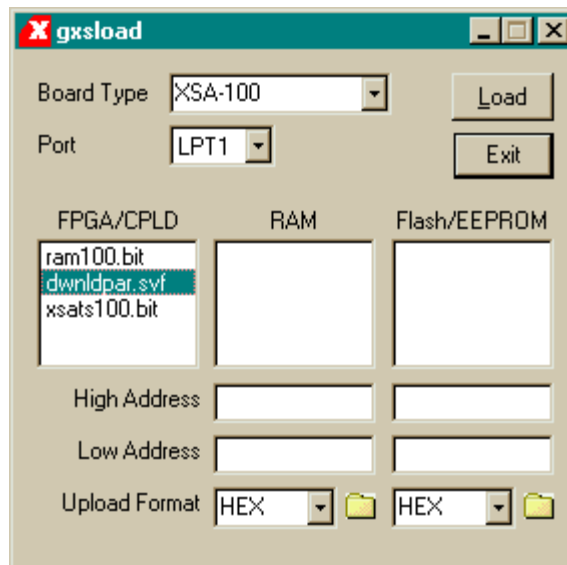
After setting the board type and parallel port, you can download .BIT or .SVF files to the Spartan-II FPGA or XC9572XL CPLD on your XSA Board simply by dragging them to the FPGA/CPLD area of the GXSLoad window as shown below.



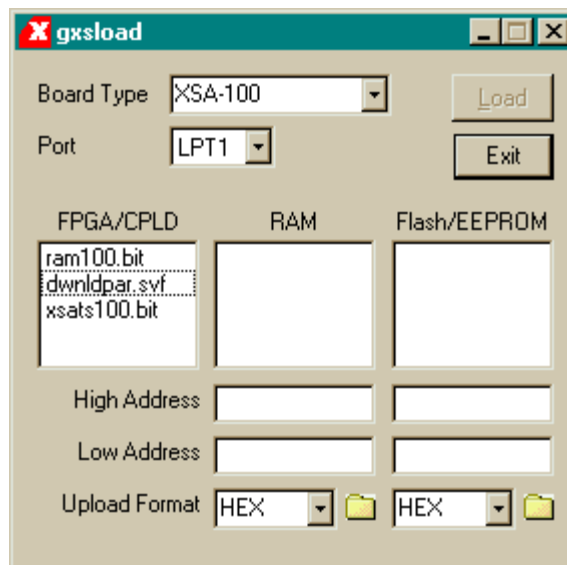
Once you release the left mouse button and drop the file, the highlighted file name appears in the FPGA/CPLD area and the Load button in the GXSLoad window is enabled. Clicking on the Load button will begin sending the highlighted file to the XSA Board through the parallel port connection. .BIT files contain configuration bitstreams that are loaded into the FPGA while .SVF files will go to the CPLD. GXSLoad will reject any non-downloadable files (ones with a suffix other than .BIT or .SVF). During the downloading process, GXSLoad will display the name of the file and the progress of the current download.



You can drag & drop multiple files into the FPGA/CPLD area. Clicking your mouse on a filename will highlight the name and select it for downloading. Only one file at a time can be selected for downloading.



Double-clicking the highlighted file will deselect it so no file will be downloaded. Doing this disables the Load button.



Storing Non-Volatile Designs in Your XSA Board

The Spartan-II FPGA on the XSA Board stores its configuration in an on-chip SRAM which is erased whenever power is removed. Once your design is finished, you may want to store the bitstream in the 256-KByte Flash device on the XSA Board which configures the FPGA for operation as soon as power is applied.

Before downloading to the Flash, the FPGA .BIT file must be converted into a .EXO or .MCS format using one of the following commands:

```
promgen -u 0 file.bit -p exo -s 256
```

```
promgen -u 0 file.bit -p mcs -s 256
```

In the commands shown above, the bitstream in the file.bit file is transformed into an .EXO or .MCS file format starting at address zero and proceeding upward until an upper limit of 256 KBytes is reached.

Before attempting to program the Flash, you must place all four DIP switches into the OFF position!

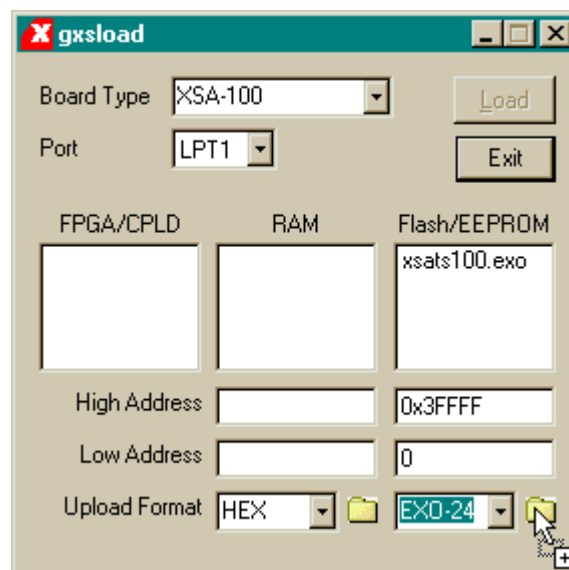
After the .EXO or .MCS file is generated, it is loaded into the Flash device by dragging it into the Flash/EEPROM area and clicking on the Load button. This activates the following sequence of steps:

1. The entire Flash device is erased.
2. The CPLD on the XSA Board is reprogrammed to create an interface between the Flash device and the PC parallel port. (This interface is stored in the finf.svf bitstream file located within the XSTOOLS\XSA folder.)
3. The contents of the .EXO or .MCS file are downloaded into the Flash through the parallel port.
4. The CPLD is reprogrammed to create a circuit that configures the FPGA with the contents of the Flash when power is applied to the XSA Board. (This configuration loader is stored in the fcng.svf bitstream file located within the XSTOOLS\XSA folder.)

Multiple files can be stored in the Flash device just by dragging them into the Flash/EEPROM area, highlighting the files to be downloaded and clicking the Load button. (Note that anything previously stored in the Flash will be erased by each new download.) This is useful if you need to store information in the Flash in addition to the FPGA bitstream. Files are selected and de-selected for downloading just by clicking on their names in the Flash/EEPROM area. **The address ranges of the data in each file should not overlap or this will corrupt the data stored in the Flash device!**

You can also examine the contents of the Flash device by uploading it to the PC. To upload data from an address range in the Flash, type the upper and lower bounds of the range into the High Address and Low Address fields below the Flash/EEPROM area, and select the format in which you would like to store the data using the Upload Format pulldown list. Then click on the file icon and drag & drop it into any folder. This activates the following sequence of steps:

1. The CPLD on the XSA Board is reprogrammed to create an interface between the Flash device and the PC parallel port.
2. The Flash data between the high and low addresses (inclusive) is uploaded through the parallel port.
3. The uploaded data is stored in a file named FLSHUPLD with an extension that reflects the file format.



The uploaded data can be stored in the following formats:

MCS: Intel hexadecimal file format. This is the same format generated by the promgen utility with the `-p mcs` option.

HEX: Identical to MCS format.

EXO-16: Motorola S-record format with 16-bit addresses (suitable for 64 KByte uploads only).

EXO-24: Motorola S-record format with 24-bit addresses. This is the same format generated by the promgen utility with the `-p exo` option.

EXO-32: Motorola S-record format with 32-bit addresses.

XESS-16: XESS hexadecimal format with 16-bit addresses. (This is a simplified file format that does not use checksums.)

XESS-24: XESS hexadecimal format with 24-bit addresses.

XESS-32: XESS hexadecimal format with 32-bit addresses.

After the data is uploaded from the Flash, the CPLD on the XSA Board is left with the Flash interface programmed into it. You will need to reprogram the CPLD with either the parallel port or Flash configuration circuit before the board will function again. The CPLD configuration bitstreams are stored in the following files:

XSTOOLS\XSA\dwlnldpar.svf: Drag & drop this file into the FPGA/CPLD area and click on the Load button to put the XSA in a mode where it will configure the FPGA through the parallel port.

XSTOOLS\XSA\fcnfg.svf: Drag & drop this file into the FPGA/CPLD area and click on the Load button to put the XSA in a mode where it will configure the FPGA with the contents of the Flash device upon power-up.

Downloading and Uploading Data to the SDRAM in Your XSA Board

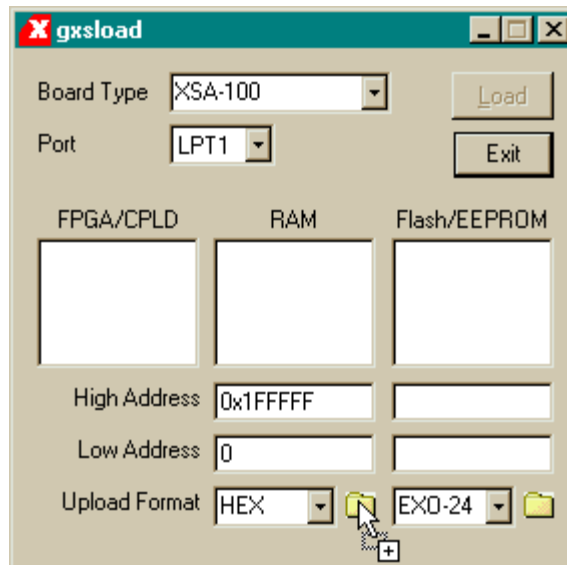
The XSA-100 Board contains a 16-MByte synchronous DRAM (8M x 16 SDRAM) whose contents can be downloaded and uploaded by GXSLOAD. (The XSA-50 has an 8-MByte SDRAM organized as 4M x 16.) This is useful for initializing the SDRAM with data for use by the FPGA and then reading the SDRAM contents after the FPGA has operated upon it. The SDRAM is loaded with data by dragging & dropping one or more .EXO, .MCS, .HEX, and/or .XES files into the RAM area of the GXSLOAD window and then clicking on the Load button. This activates the following sequence of steps:

1. The Spartan-II FPGA on the XSA Board is reprogrammed to create an interface between the RAM device and the PC parallel port. (This interface is stored in the ram100.bit or ram50.bit bitstream file located within the XSTOOLS\XSA folder. **The CPLD must have previously been loaded with the dwlnldpar.svf file found in the same folder.**)
2. The contents of the .EXO, .MCS, .HEX or .XES files are downloaded into the SDRAM through the parallel port. **The data in the files will overwrite each other if their address ranges overlap.**
3. If any file is highlighted in the FPGA/CPLD area, then this bitstream is loaded into the FPGA or CPLD on the XSA Board. Otherwise the FPGA remains configured as an interface between the PC and the SDRAM.

You can also examine the contents of the SDRAM device by uploading it to the PC. To upload data from an address range in the SDRAM, type the upper and lower bounds of the range into the High Address and Low Address fields below the RAM area, and select the format in which you would like to store the data using the Upload Format pulldown list. Then click on the file icon and drag & drop it into any folder. This activates the following sequence of steps:

1. The Spartan-II FPGA on the XSA Board is reprogrammed to create an interface between the RAM device and the PC parallel port. (This interface is stored in the ram100.bit or ram50.bit bitstream file located within the XSTOOLS\XSA folder.)
2. The SDRAM data between the high and low addresses (inclusive) is uploaded through the parallel port.

3. The uploaded data is stored in a file named RAMUPLD with an extension that reflects the file format.



The 16-bit data words in the SDRAM are mapped into the eight-bit data format of the .HEX, .MCS, .EXO and .XES files using a Big Endian style. That is, the 16-bit word at address N in the SDRAM is stored in the eight-bit file with the upper eight bits at location $2N$ and the lower eight bits at location $2N+1$. This byte-ordering applies for both RAM uploads and downloads.

4

Programmer's Models

This section describes the various sections of the XSA Board and shows how the I/O of the FPGA and CPLD are connected to the rest of the circuitry. The schematics which follow are less detailed so as to simplify the descriptions. Please refer to the complete schematics at the end of this document if you need more details.

XSA Board Organization

The XSA Board contains the following components:

XC2S50 or XC2S100 Spartan-II FPGA: This is the main repository of programmable logic on the XSA Board.

XC9572XL CPLD: This CPLD manages the interface between the PC parallel port and the rest of the XSA Board.

Osc: A programmable oscillator generates the master clock for the XSA Board.

Flash: A 128 or 256-KByte Flash device provides non-volatile storage for data and configuration bitstreams.

SDRAM: An 8 or 16-MByte SDRAM provides volatile data storage accessible by the FPGA.

LED: A seven-segment LED allows visible feedback as the XSA Board operates.

DIP switch: A four-position DIP switch passes settings to the XSA Board or controls the upper address bits of the Flash device.

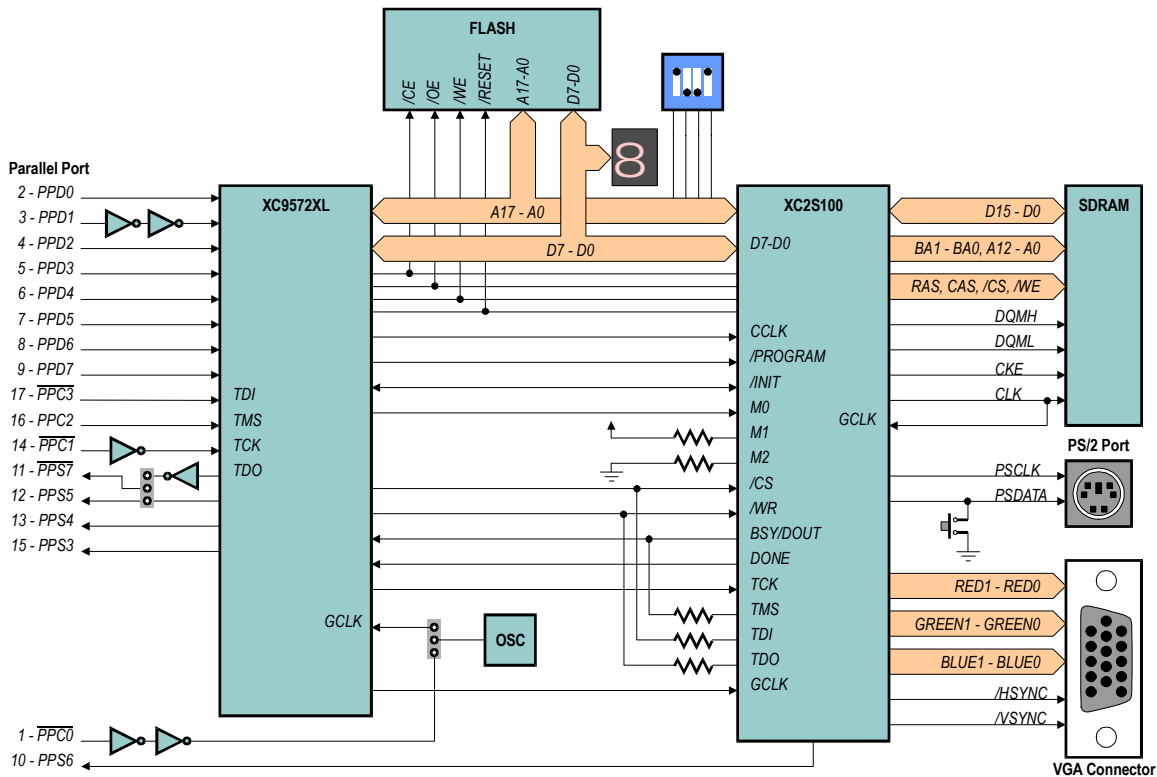
Pushbutton: A single pushbutton sends momentary contact information to the FPGA.

Parallel Port: This is the main interface for passing configuration bitstreams and data to and from the XSA Board.

PS/2 Port: A keyboard or mouse can interface to the XSA Board through this port.

VGA Port: The XSA Board can send signals to display graphics on a VGA monitor through this port.

Prototyping Header: Many of the FPGA I/O pins are connected to the 84 pins on the bottom of the XSA Board that are meant to mate with solderless breadboards.



• Figure 3: XSA Board programmer's model.

Programmable logic: Spartan-II FPGA and XC9572XL CPLD

The XSA Board contains two programmable logic chips:

- A 50-Kgate XC2S50 or 100-Kgate Xilinx XC2S100 [Spartan-II FPGA](#) in a 144-pin PQFP package. The FPGA is the main repository of programmable logic on the XSA Board.
- A Xilinx [XC9572XL CPLD](#) is used to manage the configuration of the FPGA via the parallel port. The CPLD also controls the programming of the Flash RAM on the XSA Board.

100 MHz Programmable Oscillator

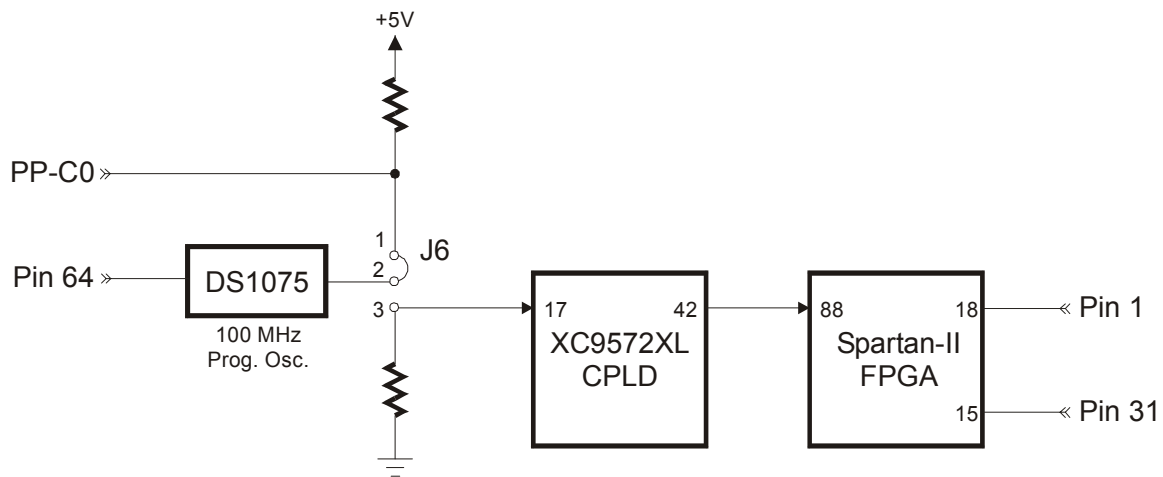
A [Dallas DS1075 programmable oscillator](#) provides a clock signal to both the FPGA and the CPLD. The DS1075 has a maximum frequency of 100 MHz that is divided to provide frequencies of 100 MHz, 50 MHz, 33.3 MHz, 25 MHz, ..., 48.7 KHz. The clock signal from the DS1075 is connected to a dedicated clock input of the CPLD. The CPLD passes the clock signal on to the FPGA. This allows the CPLD to control the clock source for the FPGA.

To set the divisor value, the DS1075 must be placed in its programming mode. This is done by pulling the clock output to +5V on power-up with a shunt across pins 1 and 2 of jumper J6. Then programming commands to set the divisor are sent to the DS1075 through control pin C0 of the parallel port. The divisor is stored in EEPROM in the DS1075 so it will be retained even when power is removed from the XSA Board.

The shunt on jumper J6 must be across pins 2 and 3 to make the oscillator output a clock signal upon power-up. The clock signal enters a dedicated clock input of the CPLD. Then the CPLD can output a clock signal to a dedicated clock input of the FPGA.

To get a precise frequency value or to sync the XSA circuitry with an external system, you can insert an external clock signal of up to 50 MHz through pin 64 of the prototyping header. This external clock takes the place of the internal 100 MHz clock source in the DS1075 oscillator. You must use the GXSETCLK software utility to enable the external clock input of the DS1075.

Clock signals can also be directly applied to two of the dedicated clock inputs of the FPGA through the pins of the prototyping header.

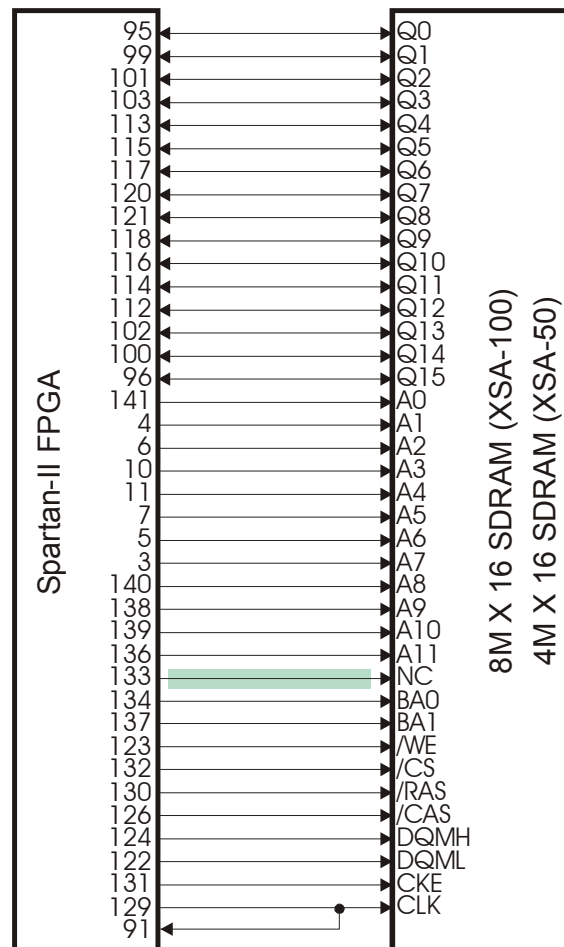


Synchronous DRAM

The various SDRAM organizations and manufacturers used on the XSA Boards are given in the following table.

Board	SDRAM	
	Organization	Manufacturer & Part No.
XSA-50	4M x 16	Hynix HY57V641620HGT-H
	4M x 16	Samsung K4S641632F-TC75000
XSA-100	8M x 16	Hynix HY57V281620HCT-H
	8M x 16	Samsung K4S281632E-TC75000

The SDRAM is connected to the FPGA as shown below. Currently, FPGA pin 133 drives a no-connect pin of the SDRAM but this could be used in the future as the thirteenth row/column address bit of a larger SDRAM. Also, the SDRAM clock signal is re-routed back to a dedicated clock input of the FPGA to allow synchronization of the FPGA's internal operations with the SDRAM operations.

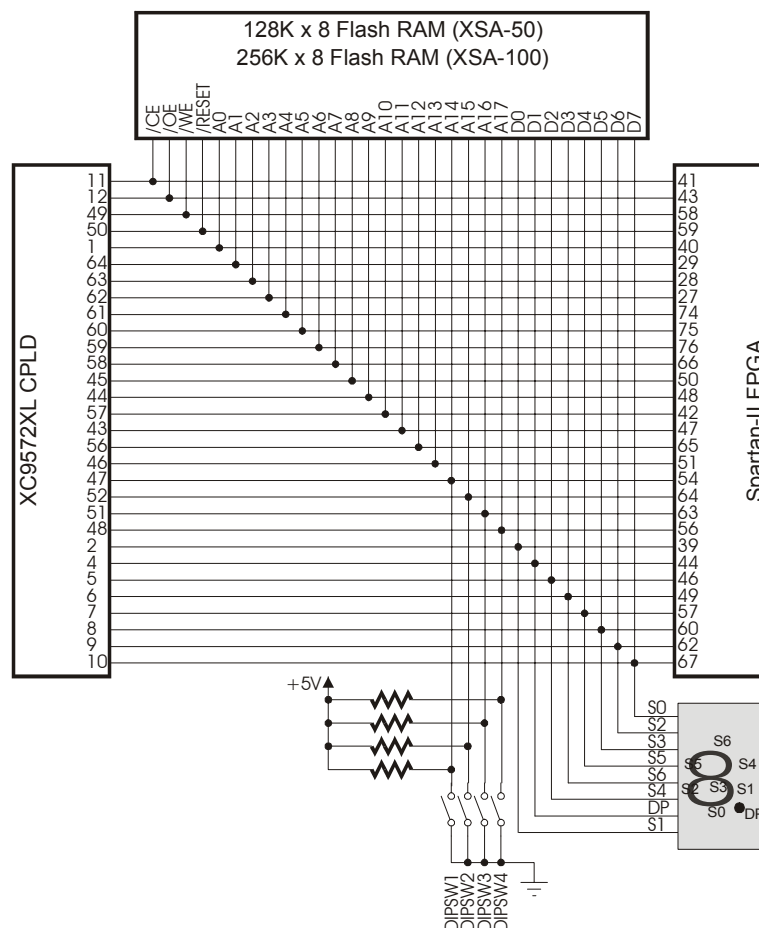


Flash RAM

The Flash RAM organizations and manufacturer used on the XSA Boards are given in the following table.

Board	Flash RAM	
	Organization	Manufacturer & Part No.
XSA-50	128K x 8	Atmel AT49F001 Flash RAM
XSA-100	256K x 8	Atmel AT49F002 Flash RAM

The Flash RAM is connected so both the FPGA and CPLD have access. Typically, the CPLD will program the Flash with data passed through the parallel port. If the data is an FPGA configuration bitstream, then the CPLD can be configured to program the FPGA with the bitstream from Flash whenever the XSA Board is powered up. (See the application note [XSA Flash Programming and SpartanII Configuration](#) for more details on this.) After power-up, the FPGA can read and/or write the Flash. (Of course, the CPLD and FPGA have to be programmed such that they do not conflict if both are trying to access the Flash.) The Flash is disabled by raising the /CE pin to a logic 1 thus making the I/O lines connected to the Flash available for general-purpose communication between the FPGA and the CPLD.



Seven-Segment LED

The XSA Board has a 7-segment LED digit for use by the FPGA or the CPLD. The segments of this LED are active-high meaning that a segment will glow when a logic-high is applied to it.

The LED shares the same pins as the eight bits of the Flash RAM data bus.

Four-Position DIP Switch

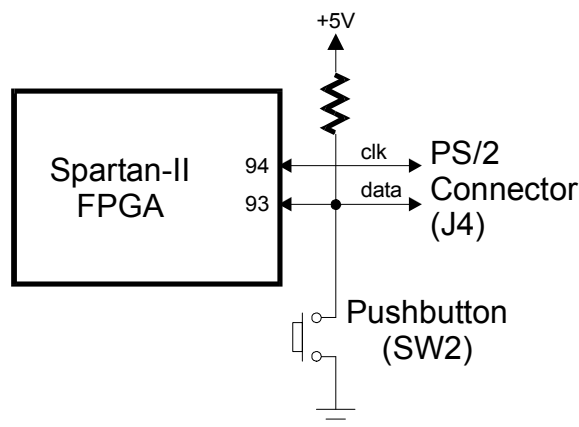
The XSA Board has a bank of four DIP switches accessible from the CPLD and FPGA. When closed or ON, each switch pulls the connected pin of the FPGA and CPLD to ground. Otherwise, the pin is pulled high through a resistor when the switch is open or OFF.

When not being used, the DIP switches should be left in the open or OFF configuration so the pins of the FPGA and CPLD are not tied to ground and can freely move between logic low and high levels.

The DIP switches also share the same pins as the uppermost four bits of the Flash RAM address bus. If the Flash RAM is programmed with several FPGA bitstreams, then the DIP switches can be used to select a particular bitstreams which will be loaded into the FPGA by the CPLD on power-up. However, this feature is not currently supported by the CPLD configuration that loads the FPGA from the Flash RAM (XSTOOLS\XSA\fcnfsg.svf).

PS/2 Port

The XSA Board provides a PS/2-style interface (mini-DIN connector J4) to either a keyboard or a mouse. The FPGA receives two signals from the PS/2 interface: a clock signal and a serial data stream that is synchronized with the falling edge of the clock.

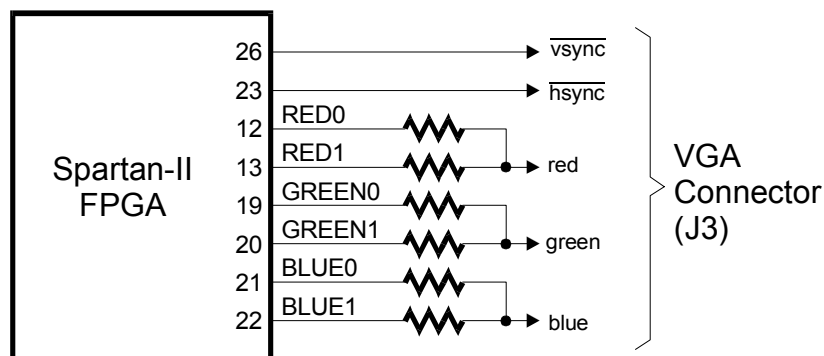


Pushbutton

The XSA Board has a single pushbutton that shares the FPGA pin connected to the data line of the PS/2 port. The pushbutton applies a low level to the FPGA pin when pressed and a resistor pulls the pin to a high level when the pushbutton is not pressed.

VGA Monitor Interface

The FPGA can generate a video signal for display on a VGA monitor. When the FPGA is generating VGA signals, the FPGA outputs two bits each of red, green, and blue color information to a simple resistor-ladder DAC. The outputs of the DAC are sent to the RGB inputs of a VGA monitor along with the horizontal and vertical sync pulses (/HSYNC, /VSYNC) from the FPGA.



Parallel Port Interface

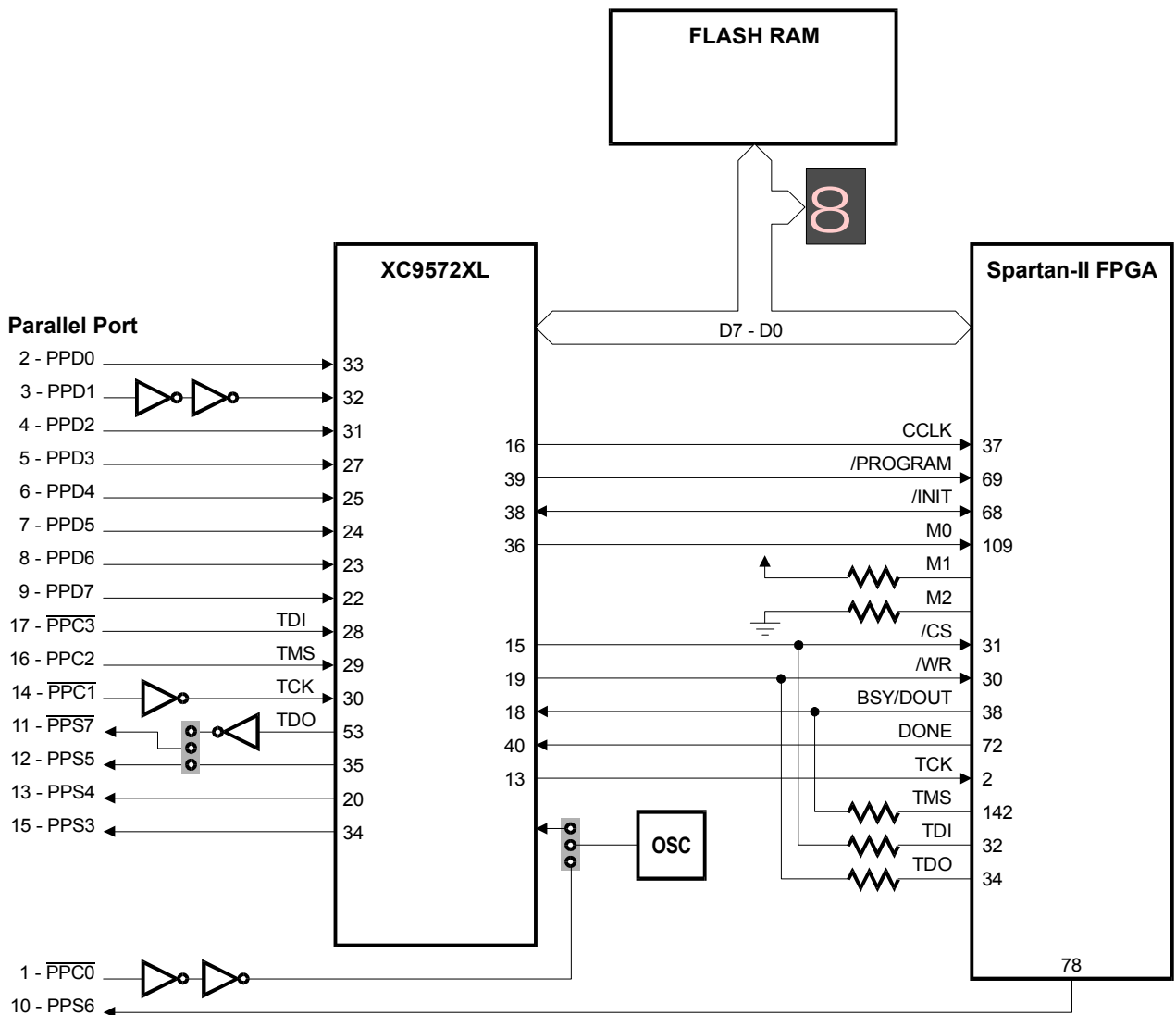
The parallel port is the main interface for communicating with the XSA Board. Control line C0 goes directly to the DS1075 oscillator and is used for setting the divisor as described previously, and status line S6 connects directly to the FPGA for use as a communication line from the FPGA back to the PC. The CPLD handles the fifteen remaining active lines of the parallel port as follows.

Three of the parallel port control lines, C1–C3, connect to the JTAG pins through which the CPLD is programmed. The C1 control line clocks configuration data presented on the C3 line into the CPLD while the C2 signal steers the actions of the CPLD programming state machine. Meanwhile, information from the CPLD returns to the PC through status line S7.

The eight data lines, D0–D7, and the remaining three status lines, S3–S5, connect to general-purpose pins of the CPLD. The CPLD can be programmed to act as an interface between the FPGA and the parallel port (the `dwndpar.svf` file is an example of such an interface). Schmitt-trigger inverters are inserted into the D1 line so it can carry a clean clock edge for use by any state machine programmed into the CPLD. The CPLD connects to the configuration pins of the Spartan-II FPGA so it can pass bitstreams from the parallel port to the FPGA. The actual configuration data is presented to the FPGA on the same 8-bit bus that also connects to the Flash RAM and seven-segment LED. The CPLD also drives the configuration pins (CCLK, /PROGRAM, /CS, and /WR) of the FPGA.

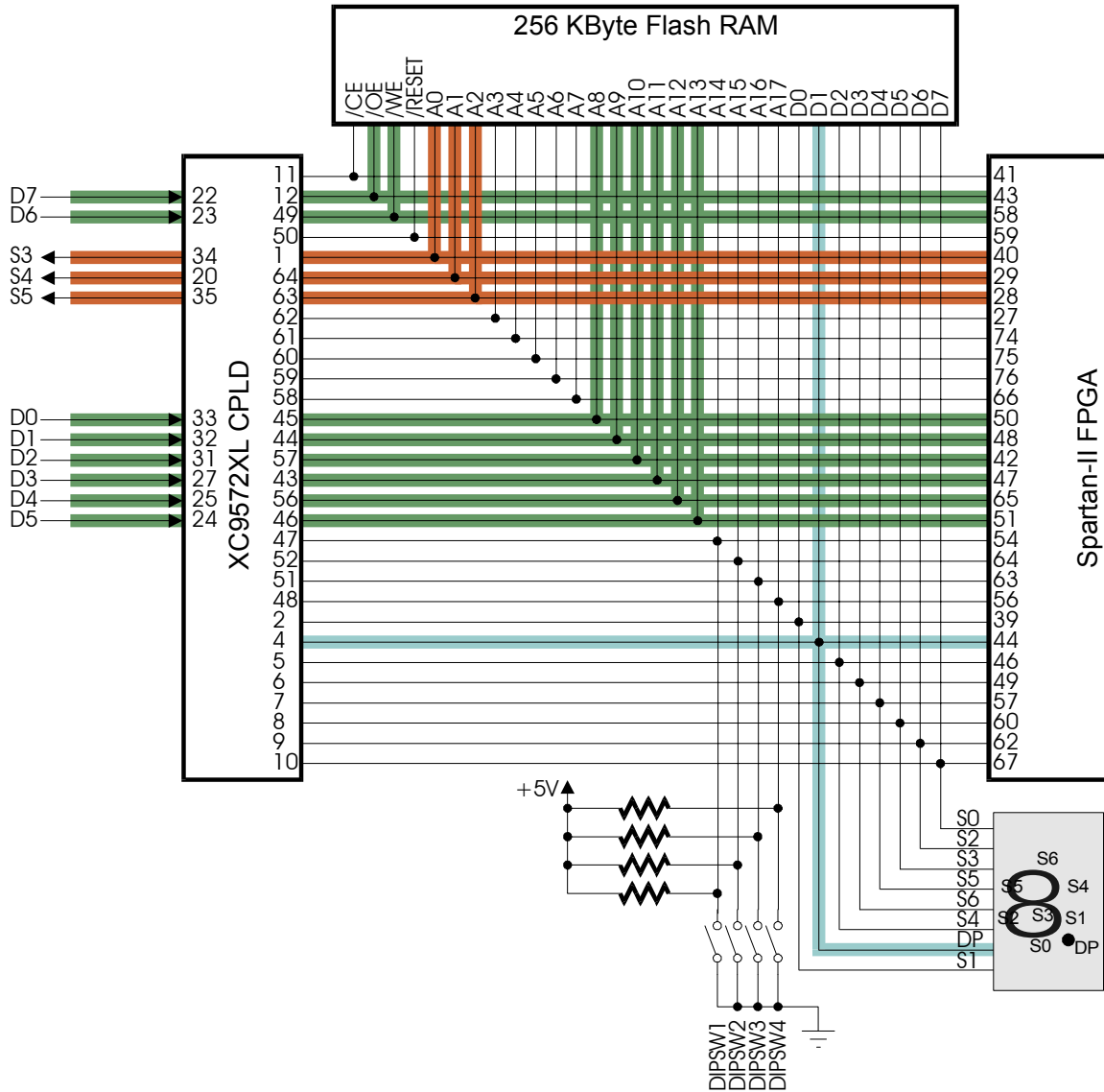
that control the loading of a bitstream. The CPLD uses the M0 input of the FPGA to select either the slave-serial or master-select configuration mode (M1 and M2 are already hard-wired to VCC and GND, respectively.) The CPLD can monitor the status of the bitstream download through the /INIT, DONE, and BSY/DOUT pins of the FPGA.

The CPLD also has access to the FPGA's JTAG pins: TCK, TMS, TDI, TDO. The TMS, TDI, and TDO pins share the connections with the BSY/DOUT, /CS, and /WR pins. With these connections, the CPLD can be programmed with an interface that allows configuration of the Spartan-II FPGA through the Xilinx iMPACT software. Jumper J9 allows the connection of status pin S7 to the general-purpose CPLD pin that also drives status pin S5. This is required by the iMPACT software so it can check for the presence of the downloading cable.



After the SpartanII FPGA is configured with a bitstream and the DONE pin goes high, the CPLD switches into a mode that connects the parallel port data and status pins to the FPGA. This lets you pass data to the FPGA over the parallel port data lines while

receiving data from the FPGA over the status lines. The connections between the FPGA and the parallel port are shown below.



The FPGA sends data back to the PC by driving logic levels onto pins 40, 29 and 28 which pass through the CPLD and onto the parallel port status lines S3, S4 and S5, respectively. Conversely, the PC sends data to the FPGA on parallel port data lines D0–D7 and the data passes through the CPLD and ends up on FPGA pins 50, 48, 42, 47, 65, 51, 58 and 43, respectively. The FPGA should never drive these pins unless it is accessing the Flash RAM otherwise the CPLD and/or the FPGA could be damaged. The CPLD can sense when the FPGA lowers the Flash RAM chip-enable and it will release the data lines so the FPGA can drive the address, output-enable and write-enable pins of the Flash RAM without contention.

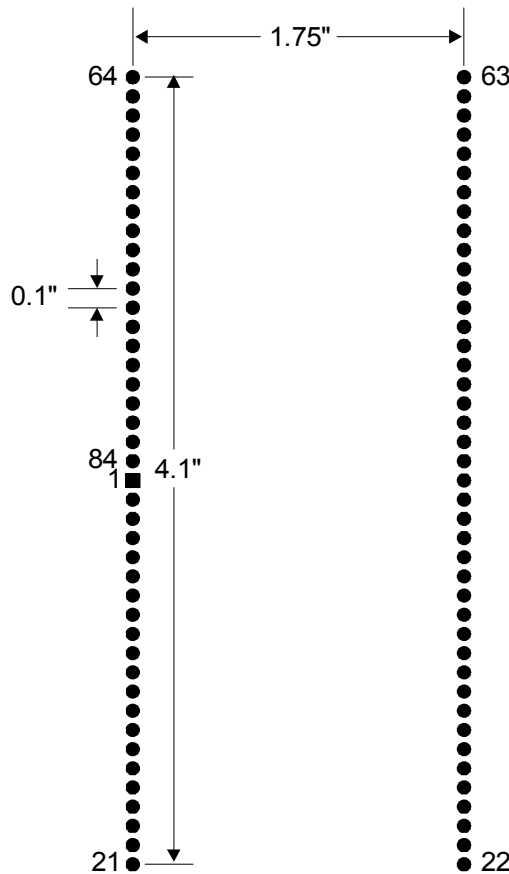
The CPLD also drives the decimal-point of the LED display to indicate when the FPGA is configured with a valid bitstream. Unless it is accessing the Flash RAM, the FPGA should never drive pin 44 to a low logic level or it may damage itself or the CPLD. But when the

FPGA lowers the Flash RAM chip-enable, the CPLD will stop driving the LED decimal-point to allow the FPGA access to data pin D1 of the Flash RAM.

For more details on how the CPLD manages the interface between the parallel port and the SpartanII FPGA both before and after device configuration, see the [XSA Parallel Port Interface application note](#).

Prototyping Header

The pins of the FPGA are accessible through the 84-pin prototyping header on the underside of the XSA Board. Pin 1 of the header (denoted by a square pad) is located in the middle of the left-hand edge of the board and the remaining 83 pins are arranged counter-clockwise around the periphery. The physical dimensions of the prototyping header and the pin arrangement are shown below.



A subset of the 144 pins on the FPGA's TQFP package connects to the prototyping header. The number of the FPGA pin connected to a given header pin is printed next to the header pin on the board. This makes it easier to find a given FPGA pin when you want to connect it to an external system. While most of the FPGA pins are already used to support functions of the XSA Board, they can also be used to interface to external systems through the prototyping header. The FPGA pins can be grouped into the various categories shown below. (Pins denoted with * are useable as general-purpose I/O; pins denoted with ** can be used as general-purpose I/O only if the CPLD interface is

reprogrammed with the alternate parallel port interface stored in the dwnldpa2.svf file; pins with no marking cannot be used as general-purpose I/O at all.)

Configuration Pins (30*, 31*, 37, 38*, 39*, 44*, 46*, 49*, 57*, 60*, 62*, 67*, 68*, 69, 72, 106, 109, 111): These pins are used to load the SpartanII FPGA with a configuration bitstream. Some of these pins are dedicated to the configuration process and cannot be used as general-purpose I/O (37, 69, 72, 106, 109, 111). The rest can be used as general-purpose I/O after the FPGA is configured. If external logic is connected to these pins, you may have to disable it during the configuration process. The DONE pin (72) can be used for this purpose since it goes to a logic high only after the configuration process is completed.

Flash RAM Pins (27*, 28*, 29*, 39*, 40*, 41*, 42**, 43**, 44*, 46*, 47**, 48**, 49*, 50**, 51**, 54*, 56*, 57*, 58**, 59*, 60*, 62*, 63*, 64*, 65**, 66*, 67*, 74*, 75*, 76*): These pins are used by the FPGA to access the Flash RAM. They can be used for general-purpose I/O under the following conditions. When the FPGA is configured from the Flash, the CPLD drives all these pins so any external logic should be disabled using the DONE pin. Also, after the configuration, the Flash chip-enable (41) should be driven high to disable the Flash RAM so it doesn't drive the data bus pins. In addition, the standard parallel port interface loaded into the CPLD (dwnldpar.svf) will drive eight of the Flash RAM pins (42, 43, 47, 48, 50, 51, 58, 65) with the logic values found on the eight data lines of the parallel port. If this is not desired, then use the alternate parallel port interface (dwnldpa2.svf) which does not drive these pins.

VGA Pins (12*, 13*, 19*, 20*, 21*, 22*, 23*, 26*): When not used to drive a VGA monitor, these pins can be used for general-purpose I/O through the prototyping header. When used as I/O, the RED0–RED1 (12–13), GREEN0–GREEN1 (19–20) and BLUE0–BLUE1 (21–22) pairs have an impedance of approximately 1 K Ω between them due to the presence of the resistor-ladder DAC circuitry.

PS/2 Pins (93*, 94*): When not used to access the PS/2 keyboard/mouse port, these pins can be used as general-purpose I/O through the prototyping header.

Global Clock Pins (15*, 18*): These pins can be used as global clock inputs or general-purpose inputs. They cannot be used as outputs.

Free Pins (77*, 78*, 79*, 80*, 83*, 84*, 85*, 86*, 87*): These pins are not connected to any other devices on the XSA Board so they can be used without restrictions as general-purpose I/O through the prototyping header.

JTAG Pins (2, 32, 34, 142): These pins are used to access the JTAG features of the FPGA. They cannot be used as general-purpose I/O pins.

A

XSA Pin Connections

The following tables list the pin numbers of the FPGA and CPLD along with the pin names of the other chips that they connect to on the XSA Board and the XStend Board. The first two tables correspond to an XSA Board + XST-2.x combination, while the last two tables correspond to an XSA Board + XST-1.x combination. Pins marked with * are useable as general-purpose I/O; pins denoted with ** can be used as general-purpose I/O only if the CPLD interface is reprogrammed with the alternate parallel port interface stored in the dwnldpa2.svf file; pins with no marking cannot be used as general-purpose I/O at all.

Connections Between the FPGA and Other XSA Board Components...												and the XST-2.x Board						
FPGA Pin	FPGA Pin Function	Net Name	CPLD Pin	Parallel Port	LEDs	Switch Button	SDRAM	Flash	VGA	PS/2	Proto. Pin	LEDs	Switch Button	SRAM	IDE Intfc.	Stereo Codec	USB	Serial Port
1	VCCO	+3.3V									PROTO54							
2	TCK	FPGA-TCK	13								PROTO16							
3 *	I/O						SDRAM-A7											
4 *	I/O						SDRAM-A1											
5 *	I/O-VREF0						SDRAM-A6											
6 *	I/O-VREF0						SDRAM-A2											
7 *	I/O						SDRAM-A5											
8	GND										PROTO52							
9	VCCINT	+2.5V									PROTO22							
10 *	I/O						SDRAM-A3											
11 *	I/O						SDRAM-A4											
12 *	I/O-VREF0								VGA-RED0		PROTO27							
13 *	I/O								VGA-RED1		PROTO28							
14	VCCINT																	
15 *	I-GCK3	FPGA-GCK3									PROTO31							
16	VCCO																	
17	GND																	
18 *	I-GCK2	FPGA-GCK2									PROTO1							
19 *	I/O								VGA-GREEN0		PROTO29							
20 *	I/O								VGA-GREEN1		PROTO32							
21 *	I/O-VREF1								VGA-BLUE0		PROTO33							
22 *	I/O								VGA-BLUE1		PROTO34							
23 *	I/O								VGA-HSYNC#		PROTO36		PUSHB4					
24	VCCINT																	
25	GND																	
26 *	I/O								VGA-VSYNC#		PROTO37		PUSHB3					
27 *	I/O-VREF1		62					FLASH-A3			PROTO50	LED2-B		RAM-A0	IDE-DMARQ			
28 *	I/O-VREF1		63	PP-S5				FLASH-A2			PROTO51	LED2-E		RAM-A10			USB-INT#	
29 *	I/O		64	PP-S4				FLASH-A1			PROTO56	LED2-G		RAM-A11			USB-SUSPEND	
30 *	I/O-WRITE#	FPGA-WR#	19								PROTO69		DIPSW1					
31 *	I/O-CS#	FPGA-CS#	15								PROTO68				IDE-RESET#			
32	TDI	FPGA-TDI	15								PROTO15							
33	GND																	
34	TDO	FPGA-TDO	19								PROTO30							
35	VCCO																	
36	VCCO																	
37	CCLK	FPGA-CCLK	16								PROTO73							
38 *	I/O-DOUT/BSY	FPGA-DOUT-BSY	18								PROTO45	LED2-DP		RAM-A1	IDE-DMACK#			
39 *	I/O-D0	FPGA-DIN-D0	2		LED-S1			FLASH-D0			PROTO71	BARLED9		RAM-A16	IDE-IORDY			
40 *	I/O		1	PP-S3				FLASH-A0			PROTO57	LED2-C		RAM-A9	IDE-INTRQ			
41 *	I/O-VREF2		11					FLASH-CE#			PROTO65							
42 **	I/O		57	PP-D2				FLASH-A10			PROTO58	LED2-F		RAM-A8	IDE-D8			
43 **	I/O-VREF2		12	PP-D7				FLASH-OE#			PROTO61			RAM-OE#	IDE-D9			
44 *	I/O-D1	FPGA-D1	4		LED-DP			FLASH-D1			PROTO40	BARLED2		RAM-D6	IDE-D1			
45	GND																	
46 *	I/O-D2	FPGA-D2	5		LED-S4			FLASH-D2			PROTO39	BARLED3		RAM-D5	IDE-D2			
47 **	I/O		43	PP-D3				FLASH-A11			PROTO59	LED2-D		RAM-A13	IDE-D10			
48 **	I/O-VREF2		44	PP-D1				FLASH-A9			PROTO60	LED2-A		RAM-A15	IDE-D11			
49 *	I/O-D3	FPGA-D3	6		LED-S6			FLASH-D3			PROTO38	BARLED4		RAM-D4	IDE-D3			
50 **	I/O		45	PP-D0				FLASH-A8			PROTO78	LED1-G		RAM-A14	IDE-D12			
51 **	I/O-IRDY		46	PP-D5				FLASH-A13			PROTO79	LED1-B		RAM-A12	IDE-D13			
52	GND																	
53	VCCO																	
54 *	I/O-TRDY		47			DIPSW1A		FLASH-A14			PROTO82	LED1-F		RAM-A7	IDE-CS0#			
55	VCCINT																	
56 *	I/O		48			DIPSW1D		FLASH-A17			PROTO83	LED1-A		RAM-A6	IDE-CS1#			
57 *	I/O-D4	FPGA-D4	7		LED-S5			FLASH-D4			PROTO35	BARLED5		RAM-D3	IDE-D4			
58 **	I/O-VREF3		49	PP-D6				FLASH-WE#			PROTO62		DIPSW2	RAM-WE#	IDE-D14			
59 *	I/O		50					FLASH-RESET#			PROTO66	BARLED10				AUDIO-LRCK		
60 *	I/O-D5	FPGA-D5	8		LED-S3			FLASH-D5			PROTO80	BARLED7		RAM-D0	IDE-D6			RS232-RD
61	GND																	
62 *	I/O-D6	FPGA-D6	9		LED-S2			FLASH-D6			PROTO81	BARLED6		RAM-D1	IDE-D5			RS232-CTS
63 *	I/O-VREF3		51			DIPSW1C		FLASH-A16			PROTO84	LED1-DP		RAM-A5	IDE-DA2			
64 *	I/O		52			DIPSW1B		FLASH-A15			PROTO3	LED1-D		RAM-A4	IDE-DA0			
65 **	I/O-VREF3		56	PP-D4				FLASH-A12			PROTO4	LED1-C		RAM-A3	IDE-D15			
66 *	I/O		58					FLASH-A7			PROTO5		DIPSW5	RAM-A2	IDE-DA1			
67 *	I/O-D7	FPGA-D7	10		LED-S0			FLASH-D7			PROTO10	BARLED8		RAM-D2	IDE-D7			
68 *	I/O-INIT#	FPGA-INIT#	38								PROTO41	BARLED1		RAM-D7	IDE-D0			
69	PROG#	FPGA-PROG#	39								PROTO55		PUSHB1					
70	VCCO																	
71	VCCO																	
72	DONE	FPGA-DONE	40								PROTO53							
73	GND																	
74 *	I/O		61					FLASH-A4			PROTO70		DIPSW3			AUDIO-SDT1		
75 *	I/O		60					FLASH-A5			PROTO77		DIPSW4			AUDIO-SCLK		
76 *	I/O		59					FLASH-A6			PROTO6	LED1-E				AUDIO-SDTO		
77 *	I/O-VREF4										PROTO9		DIPSW6			AUDIO-MCLK		
78 *	I/O			PP-S6							PROTO67		PUSHB2					
79 *	I/O-VREF4										PROTO7		DIPSW8	RAM-CE#				
80 *	I/O										PROTO8		DIPSW7					RS232-RTS
81	GND																	
82	VCCINT																	
83 *	I/O										PROTO18							RS232-TD

[illegible]

Connections Between the CPLD and Other XSA Board Components...									and the XST-2.x Board						
CPLD Pin	CPLD Pin Function	Net Name	FPGA Pin	Parallel Port	LEDs	Switch Button	Flash	Proto. Pin	LEDs	Switch Button	SRAM	IDE Intfc.	Stereo Codec	USB	Serial Port
1			40 *	PP-S3			FLASH-A0	PROTO57	LED2-C		RAM-A9	IDE-INTRQ			
2		FPGA-DIN-D0	39 *		LED-S1		FLASH-D0	PROTO71	BARLED9		RAM-A16	IDE-IORDY			
3	VCCINT														
4		FPGA-D1	44 *		LED-DP		FLASH-D1	PROTO40	BARLED2		RAM-D6	IDE-D1			
5		FPGA-D2	46 *		LED-S4		FLASH-D2	PROTO39	BARLED3		RAM-D5	IDE-D2			
6		FPGA-D3	49 *		LED-S6		FLASH-D3	PROTO38	BARLED4		RAM-D4	IDE-D3			
7		FPGA-D4	57 *		LED-S5		FLASH-D4	PROTO35	BARLED5		RAM-D3	IDE-D4			
8		FPGA-D5	60 *		LED-S3		FLASH-D5	PROTO80	BARLED7		RAM-D0	IDE-D6			RS232-RD
9		FPGA-D6	62 *		LED-S2		FLASH-D6	PROTO81	BARLED6		RAM-D1	IDE-D5			RS232-CTS
10		FPGA-D7	67 *		LED-S0		FLASH-D7	PROTO10	BARLED8		RAM-D2	IDE-D7			
11			41 *				FLASH-CE#	PROTO65							
12			43 **	PP-D7			FLASH-OE#	PROTO61			RAM-OE#	IDE-D9			
13		FPGA-TCK	2					PROTO16							
14	GND														
15	GCK1	FPGA-CS#	31 *					PROTO68				IDE-RESET#			
16	GCK2	FPGA-TDI	32					PROTO15							
17	GCK3	FPGA-CCLK	37					PROTO73							
18		PROG-OSC													
18		FPGA-DOUT-BSY	38 *					PROTO45	LED2-DP		RAM-A1	IDE-DMACK#			
18		FPGA-TMS	142					PROTO17							
19		FPGA-WR#	30 *					PROTO69		DIPSW1					
19		FPGA-TDO	34					PROTO30							
20				PPORT-S4											
21	GND														
22				PPORT-D7											
23				PPORT-D6											
24				PPORT-D5											
25				PPORT-D4											
26	VCCIO														
27				PPORT-D3											
28	TDI			PPORT-C3											
29	TMS			PPORT-C2											
30	TCK			PPORT-C1											
31				PPORT-D2											
32				PPORT-D1											
33				PPORT-D0											
34				PPORT-S3											
35				PPORT-S5											
36		FPGA-M0	109					PROTO14							
37	VCCINT														
38		FPGA-INIT#	68 *					PROTO41	BARLED1		RAM-D7	IDE-D0			
39		FPGA-PROG#	69					PROTO55		PUSHB1					
40		FPGA-DONE	72					PROTO53							
41	GND														
42		MASTER-CLK	88					PROTO13							
43			47 **	PP-D3			FLASH-A11	PROTO59	LED2-D		RAM-A13	IDE-D10			
44			48 **	PP-D1			FLASH-A9	PROTO60	LED2-A		RAM-A15	IDE-D11			
45			50 **	PP-D0			FLASH-A8	PROTO78	LED1-G		RAM-A14	IDE-D12			
46			51 **	PP-D5			FLASH-A13	PROTO79	LED1-B		RAM-A12	IDE-D13			
47			54 *			DIPSW1A	FLASH-A14	PROTO82	LED1-F		RAM-A7	IDE-CS0#			
48			56 *			DIPSW1D	FLASH-A17	PROTO83	LED1-A		RAM-A6	IDE-CS1#			
49			58 **	PP-D6			FLASH-WE#	PROTO62		DIPSW2	RAM-WE#	IDE-D14			
50			59 *				FLASH-RESET#	PROTO66	BARLED10				AUDIO-LRCK		
51			63 *			DIPSW1C	FLASH-A16	PROTO84	LED1-DP		RAM-A5	IDE-DA2			
52			64 *			DIPSW1B	FLASH-A15	PROTO3	LED1-D		RAM-A4	IDE-DA0			
53	TDO			PPORT-S7											
54	GND														
55	VCCIO														
56			65 **	PP-D4			FLASH-A12	PROTO4	LED1-C		RAM-A3	IDE-D15			
57			42 **	PP-D2			FLASH-A10	PROTO58	LED2-F		RAM-A8	IDE-D8			
58			66 *				FLASH-A7	PROTO5		DIPSW5	RAM-A2	IDE-DA1			
59			76 *				FLASH-A6	PROTO6	LED1-E				AUDIO-SDTO		
60			75 *				FLASH-A5	PROTO77		DIPSW4			AUDIO-SCLK		
61			74 *				FLASH-A4	PROTO70		DIPSW3			AUDIO-SDTI		
62			27 *				FLASH-A3	PROTO50	LED2-B		RAM-A0	IDE-DMARQ			
63			28 *	PP-S5			FLASH-A2	PROTO51	LED2-E		RAM-A10			USB-INT#	
64			29 *	PP-S4			FLASH-A1	PROTO56	LED2-G		RAM-A11			USB-SUSPEND	

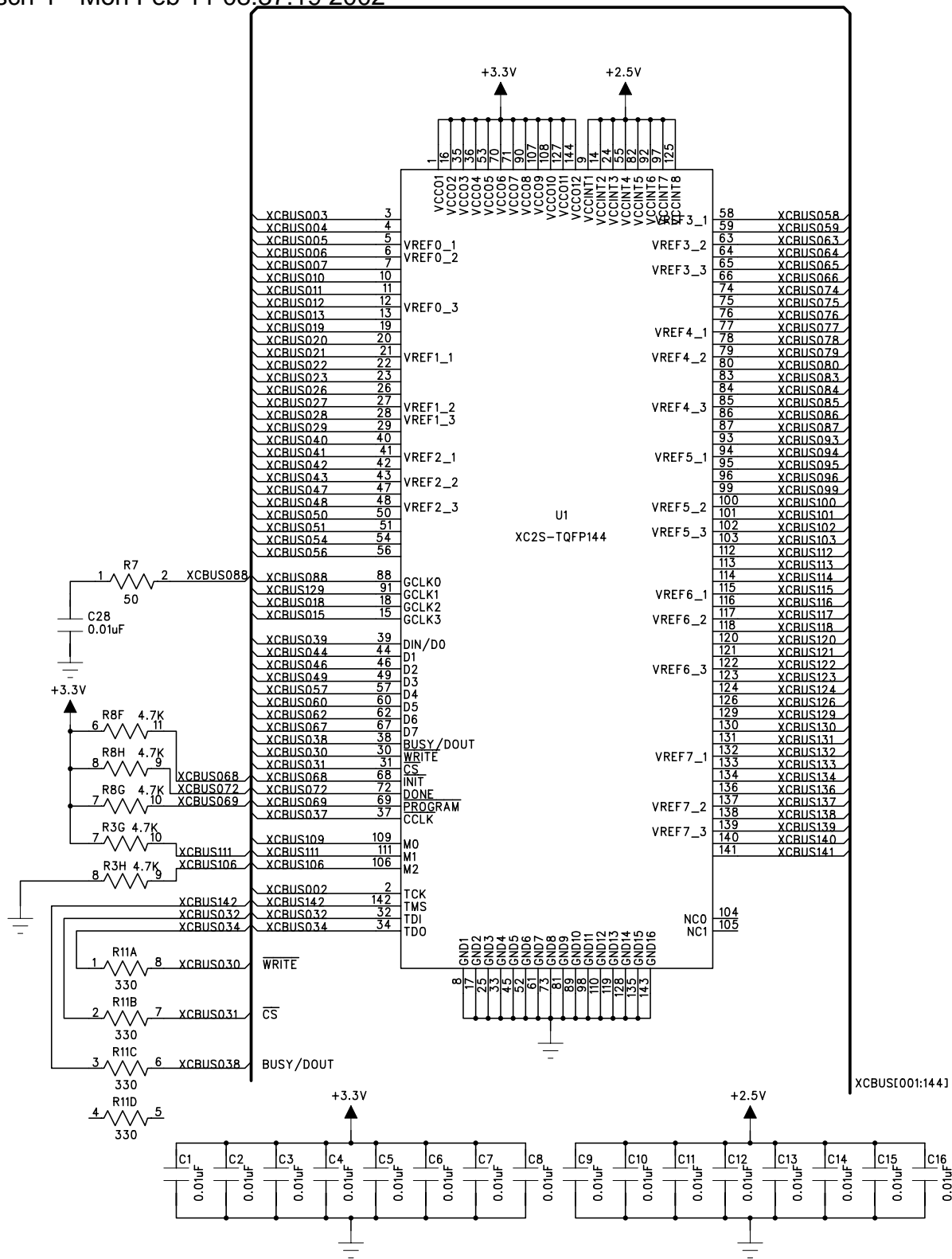
Connections Between the FPGA and Other XSA Board Components...												and the XST-1.x Board						
FPGA Pin	FPGA Pin Function	Net Name	CPLD Pin	Parallel Port	LEDs	Switch Button	SDRAM	Flash	VGA	PS/2	Proto. Pin	LEDs	Switch Button	SRAM	VGA	Stereo Codec	PS/2	Xchecker
1	VCCO	+3.3V									PROTO54							
2	TCK	FPGA-TCK	13								PROTO16							XCHK-TCK
3 *	I/O						SDRAM-A7											
4 *	I/O						SDRAM-A1											
5 *	I/O-VREF0						SDRAM-A6											
6 *	I/O-VREF0						SDRAM-A2											
7 *	I/O						SDRAM-A5											
8	GND										PROTO52							
9	VCCINT	+2.5V									PROTO22							
10 *	I/O						SDRAM-A3											
11 *	I/O						SDRAM-A4											
12 *	I/O-VREF0								VGA-RED0		PROTO27							
13 *	I/O								VGA-RED1		PROTO28	RLED-DP#		RAM-A15				
14	VCCINT																	
15 *	I-GCK3	FPGA-GCK3									PROTO31							
16	VCCO																	
17	GND																	
18 *	I-GCK2	FPGA-GCK2									PROTO1							
19 *	I/O								VGA-GREEN0		PROTO29							
20 *	I/O								VGA-GREEN1		PROTO32							XCHK-RT
21 *	I/O-VREF1								VGA-BLUE0		PROTO33							
22 *	I/O								VGA-BLUE1		PROTO34							
23 *	I/O								VGA-HSYNC#		PROTO36							
24	VCCINT																	
25	GND																	
26 *	I/O								VGA-VSYNC#		PROTO37		PUSH-RESET#					
27 *	I/O-VREF1		62					FLASH-A3			PROTO50	RLED-S4#		RAM-A12				
28 *	I/O-VREF1		63	PP-S5				FLASH-A2			PROTO51	RLED-S2#		RAM-A10				
29 *	I/O		64	PP-S4				FLASH-A1			PROTO56	RLED-S3#		RAM-A11				
30 *	I/O-WRITE#	FPGA-WR#	19								PROTO69		DIPSW8			X-PS2-DATA		
31 *	I/O-CS#	FPGA-CS#	15								PROTO68					X-PS2-CLK		
32	TDI	FPGA-TDI	15								PROTO15							XCHK-TDI
33	GND																	
34	TDO	FPGA-TDO	19								PROTO30							XCHK-RD
35	VCCO																	
36	VCCO																	
37	CCLK	FPGA-CCLK	16								PROTO73							XCHK-CCLK
38 *	I/O-DOUT/BSY	FPGA-DOUT-BSY	18								PROTO45							
39 *	I/O-D0	FPGA-DIN-D0	2		LED-S1			FLASH-D0			PROTO71							XCHK-DIN
40 *	I/O		1	PP-S3				FLASH-A0			PROTO57	RLED-S1#		RAM-A9				
41 *	I/O-VREF2		11					FLASH-CE#			PROTO65			RAM-CE#				
42 **	I/O		57	PP-D2				FLASH-A10			PROTO58	RLED-S5#		RAM-A13				
43 **	I/O-VREF2		12	PP-D7				FLASH-OE#			PROTO61			RAM-OE#				
44 *	I/O-D1	FPGA-D1	4		LED-DP			FLASH-D1			PROTO40	BARLED2		RAM-D1				
45	GND																	
46 *	I/O-D2	FPGA-D2	5		LED-S4			FLASH-D2			PROTO39	BARLED3		RAM-D2				
47 **	I/O		43	PP-D3				FLASH-A11			PROTO59	RLED-S0#		RAM-A8				
48 **	I/O-VREF2		44	PP-D1				FLASH-A9			PROTO60	RLED-S6#		RAM-A14				
49 *	I/O-D3	FPGA-D3	6		LED-S6			FLASH-D3			PROTO38	BARLED4		RAM-D3				
50 **	I/O		45	PP-D0				FLASH-A8			PROTO78	LLED-S3#		RAM-A3				
51 **	I/O-IRDY		46	PP-D5				FLASH-A13			PROTO79	LLED-S4#		RAM-A4				
52	GND																	
53	VCCO																	
54 *	I/O-TRDY		47			DIPSW1A		FLASH-A14			PROTO82	LLED-S5#		RAM-A5				
55	VCCINT																	
56 *	I/O		48			DIPSW1D		FLASH-A17			PROTO83	LLED-S6#		RAM-A6				
57 *	I/O-D4	FPGA-D4	7		LED-S5			FLASH-D4			PROTO35	BARLED5		RAM-D4				
58 **	I/O-VREF3		49	PP-D6				FLASH-WE#			PROTO62			RAM-WE#				
59 *	I/O		50					FLASH-RESET#			PROTO66		DIPSW7			CODEC-LRCK		
60 *	I/O-D5	FPGA-D5	8		LED-S3			FLASH-D5			PROTO80	BARLED7		RAM-D6				
61	GND																	
62 *	I/O-D6	FPGA-D6	9		LED-S2			FLASH-D6			PROTO81	BARLED6		RAM-D5				
63 *	I/O-VREF3		51			DIPSW1C		FLASH-A16			PROTO84	LLED-DP#		RAM-A7				
64 *	I/O		52			DIPSW1B		FLASH-A15			PROTO3	LLED-S0#		RAM-A0				
65 **	I/O-VREF3		56	PP-D4				FLASH-A12			PROTO4	LLED-S1#		RAM-A1				
66 *	I/O		58					FLASH-A7			PROTO5	LLED-S2#		RAM-A2				
67 *	I/O-D7	FPGA-D7	10		LED-S0			FLASH-D7			PROTO10	BARLED8		RAM-D7				
68 *	I/O-INIT#	FPGA-INIT#	38								PROTO41	BARLED1		RAM-D0				XCHK-INIT#
69	PROG#	FPGA-PROG#	39								PROTO55		PUSH-PROG#					XCHK-PROG#
70	VCCO																	
71	VCCO																	
72	DONE	FPGA-DONE	40								PROTO53							XCHK-DONE
73	GND																	
74 *	I/O		61					FLASH-A4			PROTO70		DIPSW6			CODEC-SDIN		
75 *	I/O		60					FLASH-A5			PROTO77		DIPSW5			CODEC-SCLK		
76 *	I/O		59					FLASH-A6			PROTO6		DIPSW4			CODEC-SDOUT		
77 *	I/O-VREF4										PROTO9		DIPSW3			CODEC-MCLK		XCHK-CLKO
78 *	I/O			PP-S6							PROTO67		PUSH-SPARE#		X-VGA-VSYNC#			
79 *	I/O-VREF4										PROTO7		DIPSW1	RAM-LCE#				XCHK-TRIG
80 *	I/O										PROTO8		DIPSW2	RAM-RCE#				XCHK-RST
81	GND																	
82	VCCINT																	
83 *	I/O										PROTO18				X-VGA-RED1			
84 *	I/O										PROTO19				X-VGA-HSYNC#			
85 *	I/O-VREF4										PROTO20				X-VGA-GREEN1			

Connections Between the CPLD and Other XSA Board Components...									and the XST-1.x Board					
CPLD Pin	CPLD Pin Function	Net Name	FPGA Pin	Parallel Port	LEDs	Switch Button	Flash	Proto. Pin	LEDs	Switch Button	SRAM	Stereo Codec	PS/2	Xchecker
1			40 *	PP-S3			FLASH-A0	PROTO57	RLED-S1#		RAM-A9			
2		FPGA-DIN-D0	39 *		LED-S1		FLASH-D0	PROTO71						XCHK-DIN
3	VCCINT													
4		FPGA-D1	44 *		LED-DP		FLASH-D1	PROTO40	BARLED2		RAM-D1			
5		FPGA-D2	46 *		LED-S4		FLASH-D2	PROTO39	BARLED3		RAM-D2			
6		FPGA-D3	49 *		LED-S6		FLASH-D3	PROTO38	BARLED4		RAM-D3			
7		FPGA-D4	57 *		LED-S5		FLASH-D4	PROTO35	BARLED5		RAM-D4			
8		FPGA-D5	60 *		LED-S3		FLASH-D5	PROTO80	BARLED7		RAM-D6			
9		FPGA-D6	62 *		LED-S2		FLASH-D6	PROTO81	BARLED6		RAM-D5			
10		FPGA-D7	67 *		LED-S0		FLASH-D7	PROTO10	BARLED8		RAM-D7			
11			41 *				FLASH-CE#	PROTO65			RAM-CE#			
12			43 **	PP-D7			FLASH-OE#	PROTO61			RAM-OE#			
13		FPGA-TCK	2					PROTO16						XCHK-TCK
14	GND													
15	GCK1	FPGA-CS#	31 *					PROTO68					X-PS2-CLK	
16	GCK2	FPGA-TDI	32					PROTO15						XCHK-TDI
17	GCK3	FPGA-CCLK	37					PROTO73						XCHK-CCLK
18		PROG-OSC												
18		FPGA-DOUT-BSY	38 *					PROTO45						
18		FPGA-TMS	142					PROTO17						XCHK-TMS
19		FPGA-WR#	30 *					PROTO69		DIPSW8			X-PS2-DATA	
19		FPGA-TDO	34					PROTO30						XCHK-RD
20				PPORT-S4										
21	GND													
22				PPORT-D7										
23				PPORT-D6										
24				PPORT-D5										
25				PPORT-D4										
26	VCCIO													
27				PPORT-D3										
28	TDI			PPORT-C3										
29	TMS			PPORT-C2										
30	TCK			PPORT-C1										
31				PPORT-D2										
32				PPORT-D1										
33				PPORT-D0										
34				PPORT-S3										
35				PPORT-S5										
36		FPGA-M0	109					PROTO14						
37	VCCINT													
38		FPGA-INIT#	68 *					PROTO41	BARLED1		RAM-D0			XCHK-INIT#
39		FPGA-PROG#	69					PROTO55		PUSH-PROG#				XCHK-PROG#
40		FPGA-DONE	72					PROTO53						XCHK-DONE
41	GND													
42		MASTER-CLK	88					PROTO13						XCHK-CLKI
43			47 **	PP-D3			FLASH-A11	PROTO59	RLED-S0#		RAM-A8			
44			48 **	PP-D1			FLASH-A9	PROTO60	RLED-S6#		RAM-A14			
45			50 **	PP-D0			FLASH-A8	PROTO78	LLED-S3#		RAM-A3			
46			51 **	PP-D5			FLASH-A13	PROTO79	LLED-S4#		RAM-A4			
47			54 *			DIPSW1A	FLASH-A14	PROTO82	LLED-S5#		RAM-A5			
48			56 *			DIPSW1D	FLASH-A17	PROTO83	LLED-S6#		RAM-A6			
49			58 **	PP-D6			FLASH-WE#	PROTO62			RAM-WE#			
50			59 *				FLASH-RESET#	PROTO66		DIPSW7		CODEC-LRCK		
51			63 *			DIPSW1C	FLASH-A16	PROTO84	LLED-DP#		RAM-A7			
52			64 *			DIPSW1B	FLASH-A15	PROTO3	LLED-S0#		RAM-A0			
53	TDO			PPORT-S7										
54	GND													
55	VCCIO													
56			65 **	PP-D4			FLASH-A12	PROTO4	LLED-S1#		RAM-A1			
57			42 **	PP-D2			FLASH-A10	PROTO58	RLED-S5#		RAM-A13			
58			66 *				FLASH-A7	PROTO5	LLED-S2#		RAM-A2			
59			76 *				FLASH-A6	PROTO6		DIPSW4		CODEC-SDOUT		
60			75 *				FLASH-A5	PROTO77		DIPSW5		CODEC-SCLK		
61			74 *				FLASH-A4	PROTO70		DIPSW6		CODEC-SDIN		
62			27 *				FLASH-A3	PROTO50	RLED-S4#		RAM-A12			
63			28 *	PP-S5			FLASH-A2	PROTO51	RLED-S2#		RAM-A10			
64			29 *	PP-S4			FLASH-A1	PROTO56	RLED-S3#		RAM-A11			

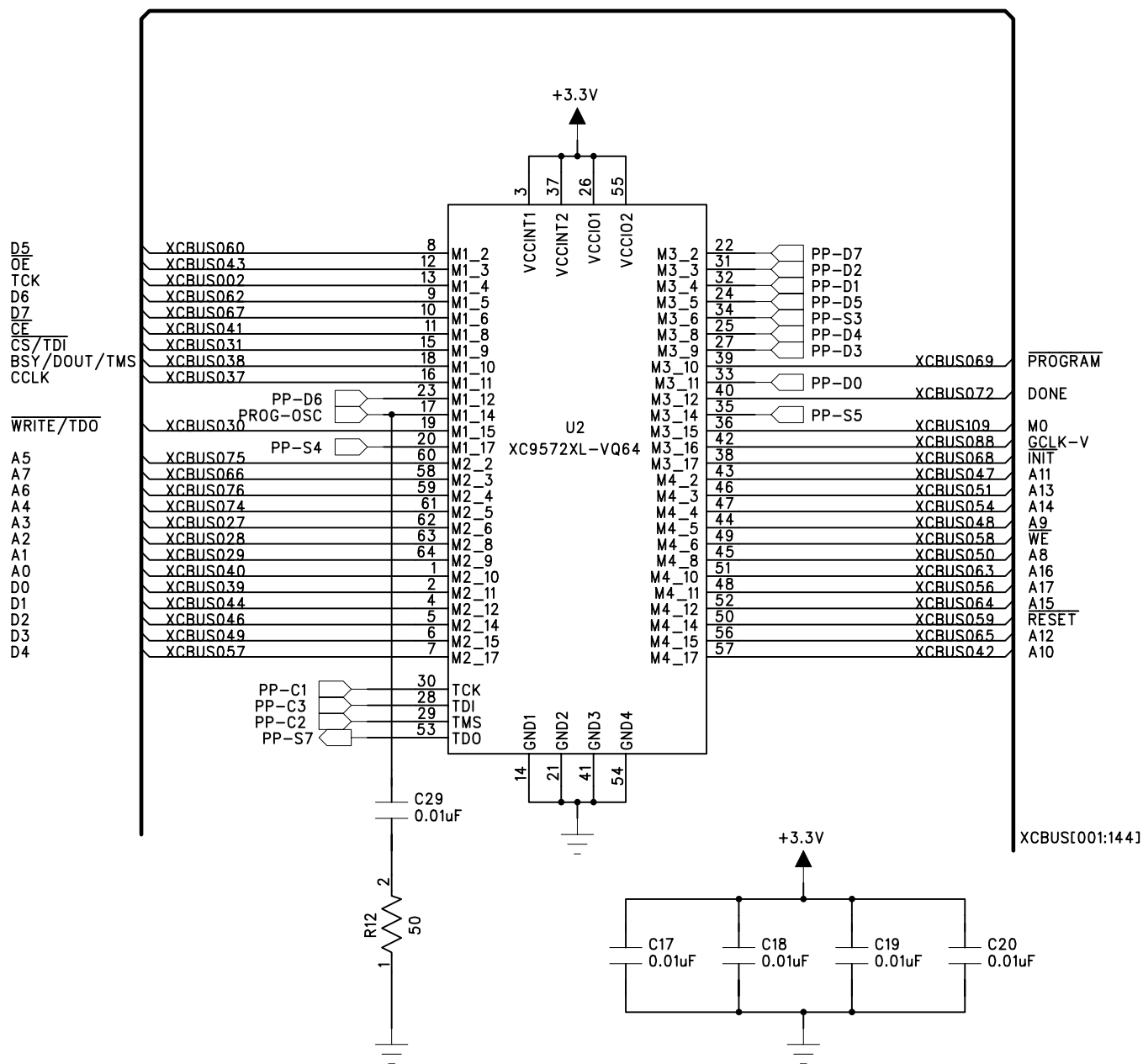


XSA Schematics

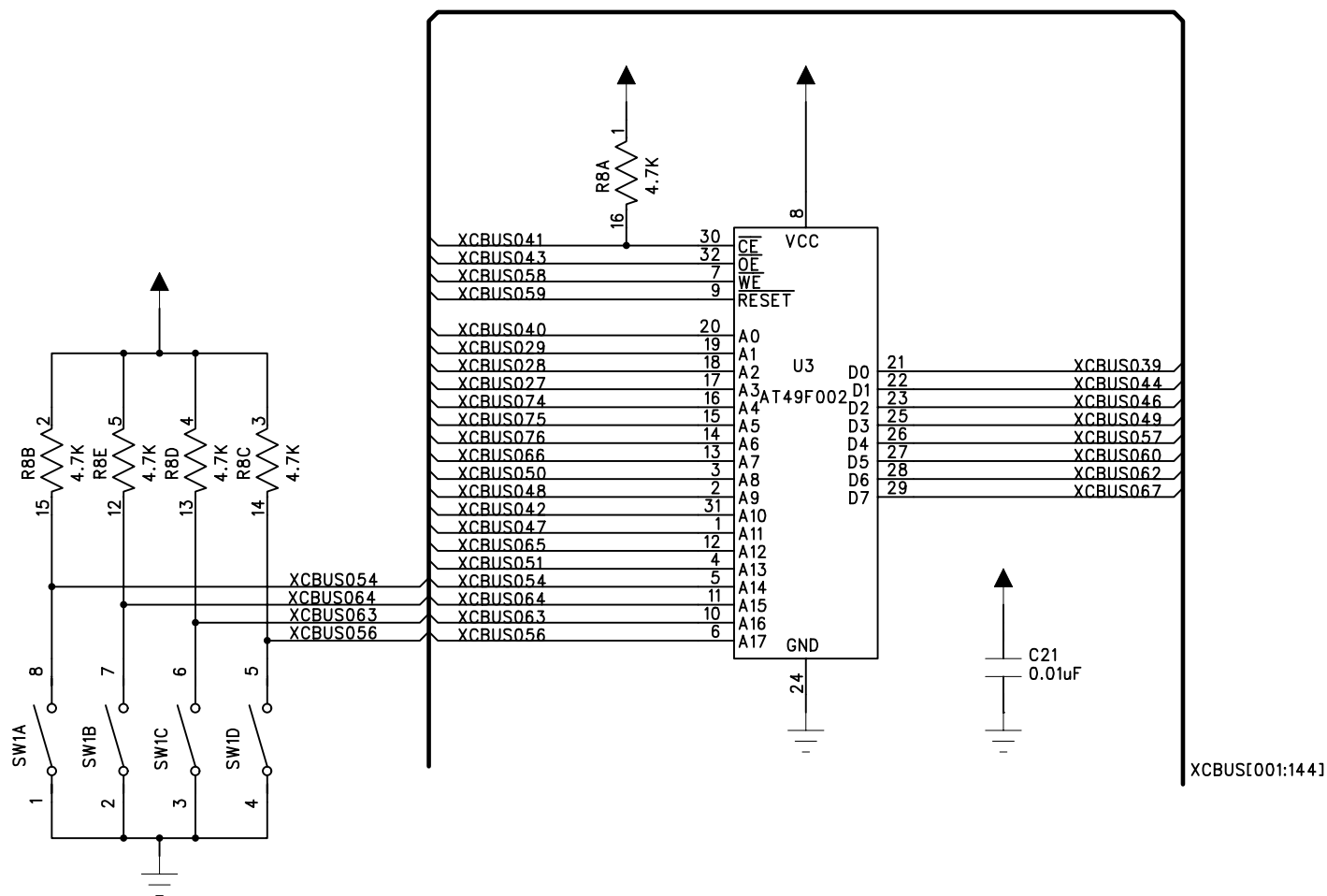
The following pages show the detailed schematics for the XSA Board.



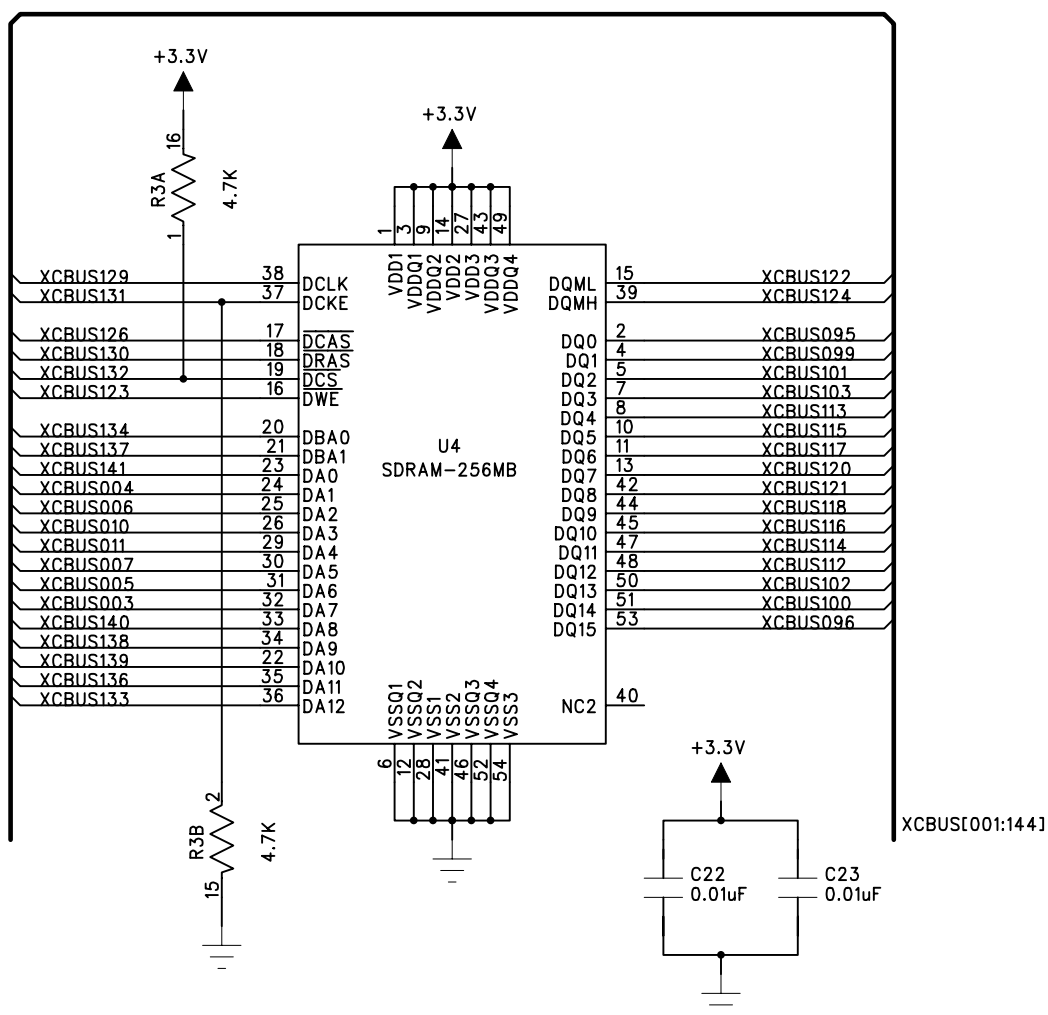
COMPANY: XESS Corporation		
TITLE: XSA Board Spartan FPGA		
DRAWN:	DATED:	REV: V1.2
RELEASED:	DATED:	SHEET: OF



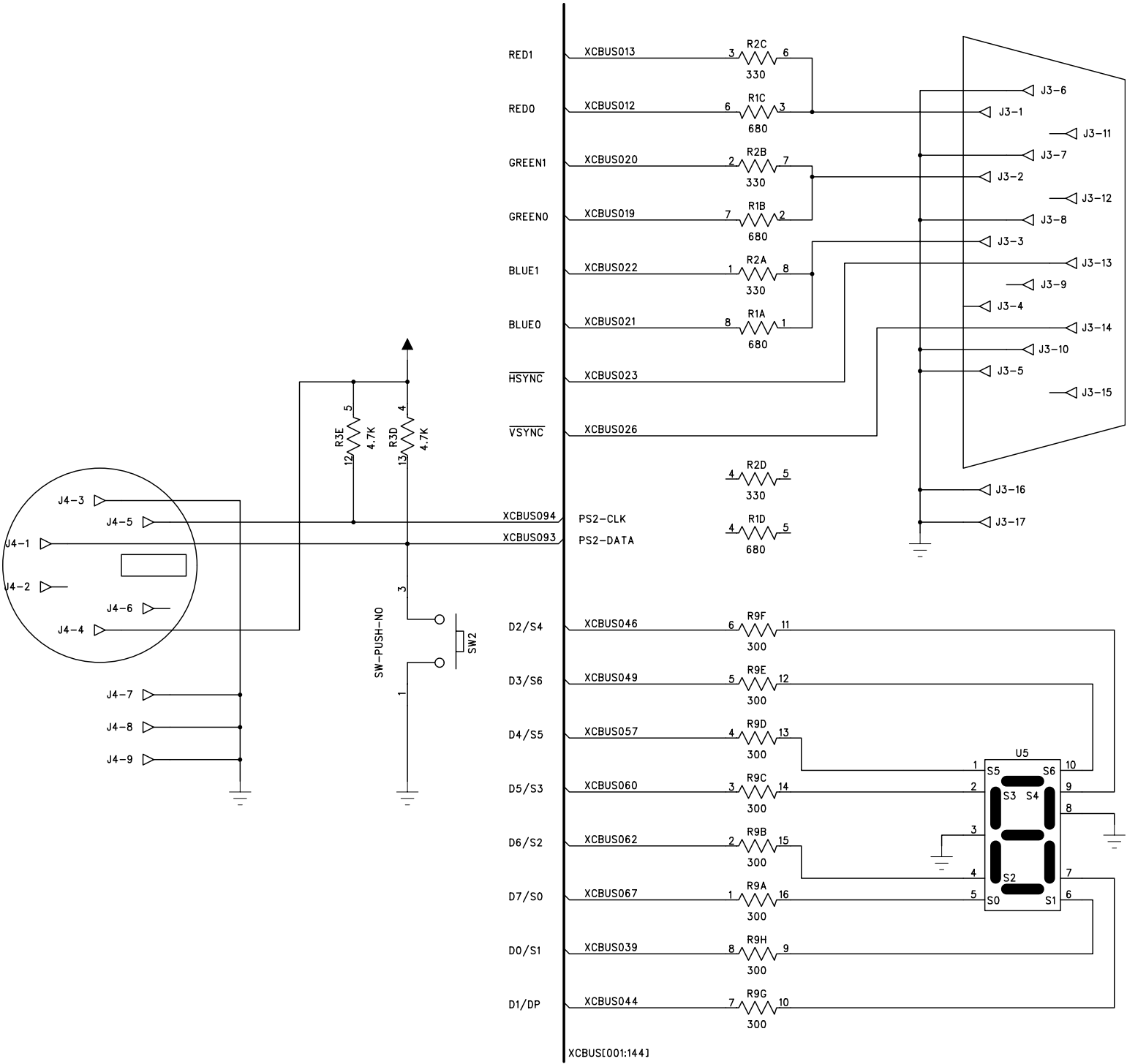
COMPANY: XESS Corporation		
TITLE: XSA Board CPLD Interface		
DRAWN:	DATED:	REV: V1.2
RELEASED:	DATED:	SHEET: OF



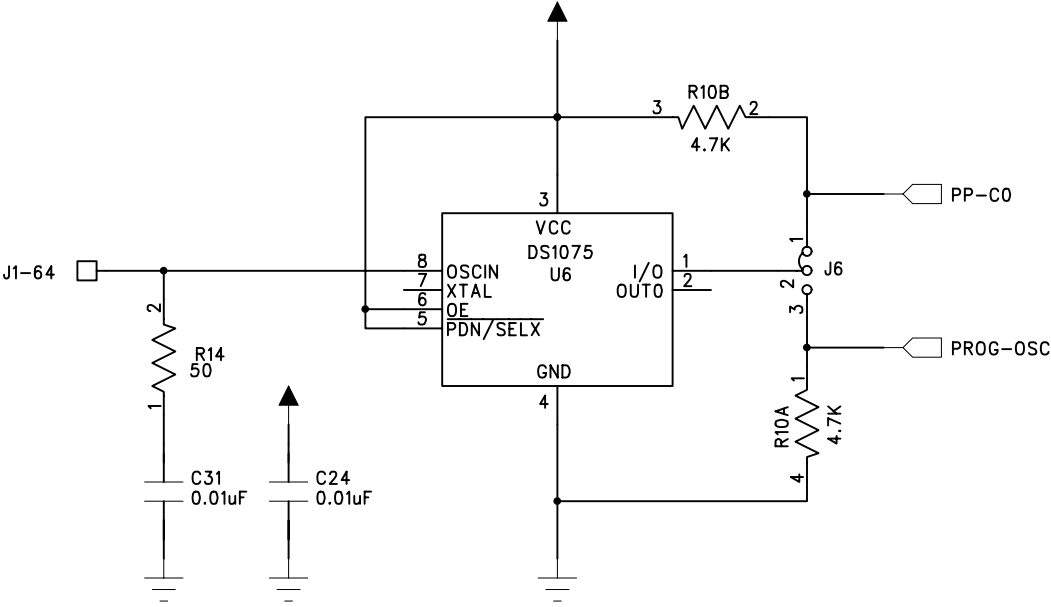
COMPANY: XESS Corporation		
TITLE: XSA Board Flash RAM		
DRAWN:	DATED:	REV: V1.2
RELEASED:	DATED:	SHEET: OF



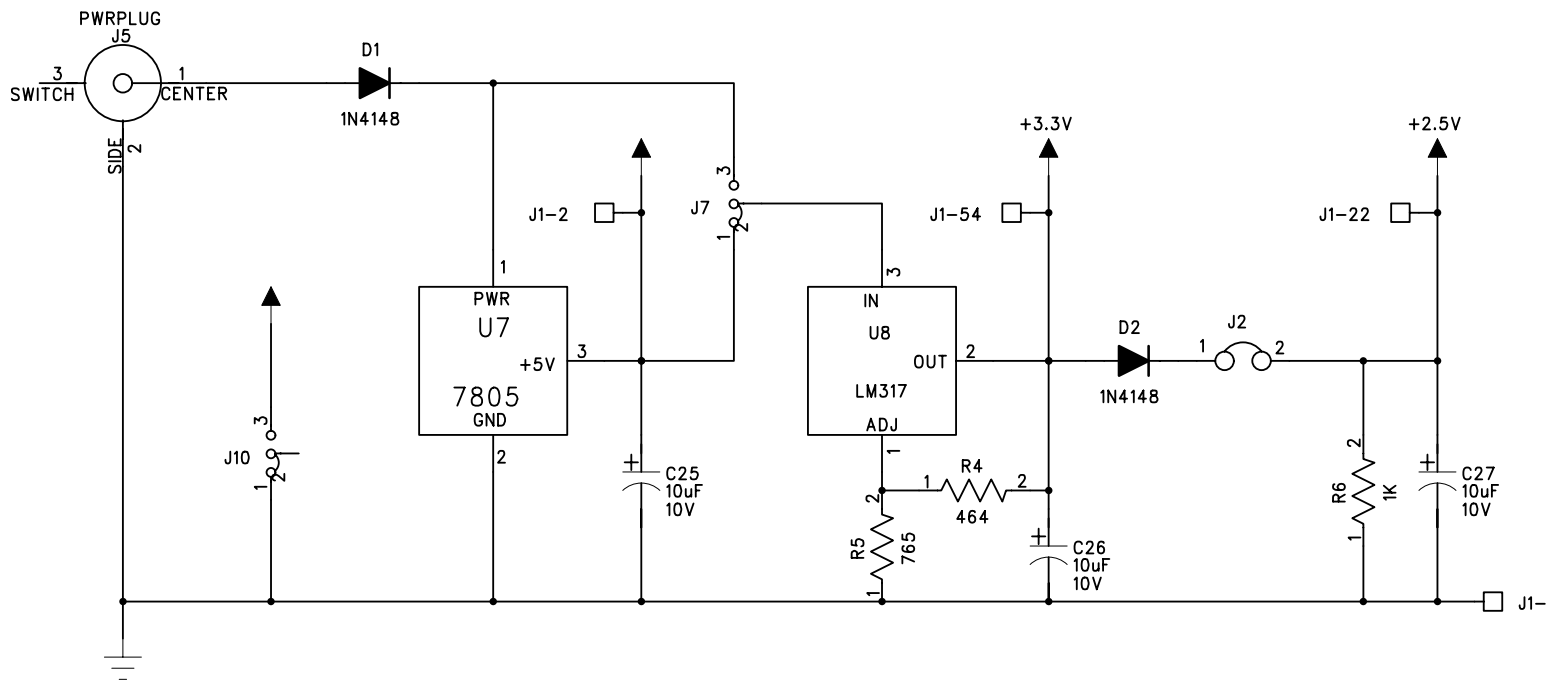
COMPANY:			XESS Corporation		
TITLE:			XSA Board Sync. DRAM		
DRAWN:		DATED:		REV: V1.2	
RELEASED:		DATED:		SHEET: OF	



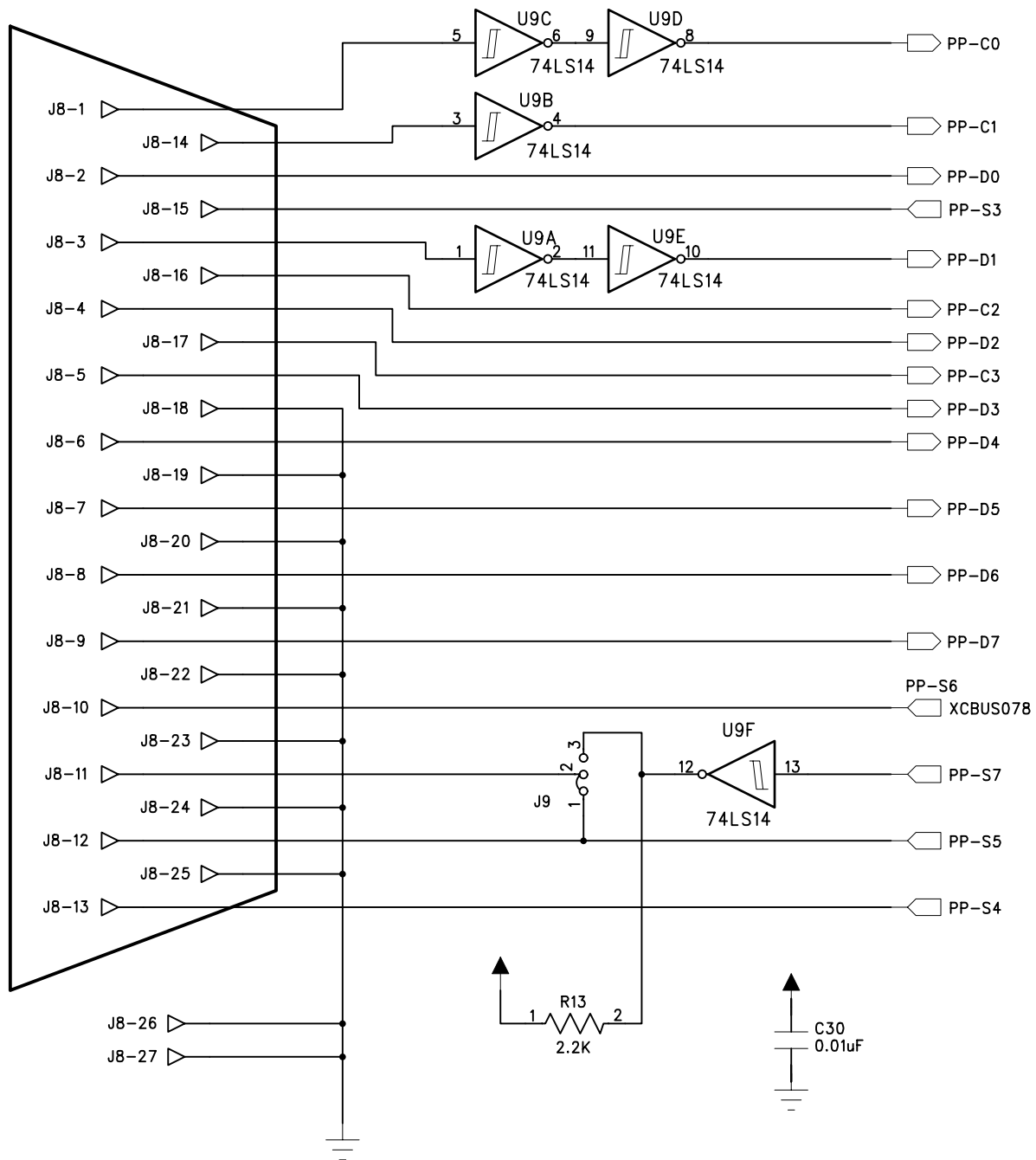
COMPANY: XESS Corporation		
TITLE: XSA Board PS/2 Port, VGA Port, LED		
DRAWN:	DATED:	REV: V1.2
RELEASED:	DATED:	SHEET: OF



COMPANY: XESS Corporation		
TITLE: XSA Board Programmable Oscillator		
DRAWN:	DATED:	REV: V1.2
RELEASED:	DATED:	SHEET: 0F



COMPANY:			XESS Corporation		
TITLE:			XSA Board Regulated Power Supplies		
DRAWN:		DATED:		REV: V1.2	
RELEASED:		DATED:		SHEET: OF	



COMPANY: XESS Corporation		
TITLE: XSA Board Parallel Port Interface		
DRAWN:	DATED:	REV: V1.2
RELEASED:	DATED:	SHEET: OF

XCBUS002	<input type="checkbox"/>	J1-16	XCBUS063	<input type="checkbox"/>	J1-84
XCBUS012	<input type="checkbox"/>	J1-27	XCBUS064	<input type="checkbox"/>	J1-3
XCBUS013	<input type="checkbox"/>	J1-28	XCBUS065	<input type="checkbox"/>	J1-4
XCBUS015	<input type="checkbox"/>	J1-31	XCBUS066	<input type="checkbox"/>	J1-5
XCBUS018	<input type="checkbox"/>	J1-1	XCBUS067	<input type="checkbox"/>	J1-10
XCBUS019	<input type="checkbox"/>	J1-29	XCBUS068	<input type="checkbox"/>	J1-41
XCBUS020	<input type="checkbox"/>	J1-32	XCBUS069	<input type="checkbox"/>	J1-55
XCBUS021	<input type="checkbox"/>	J1-33	XCBUS072	<input type="checkbox"/>	J1-53
XCBUS022	<input type="checkbox"/>	J1-34	XCBUS074	<input type="checkbox"/>	J1-70
XCBUS023	<input type="checkbox"/>	J1-36	XCBUS075	<input type="checkbox"/>	J1-77
XCBUS026	<input type="checkbox"/>	J1-37	XCBUS076	<input type="checkbox"/>	J1-6
XCBUS027	<input type="checkbox"/>	J1-50	XCBUS077	<input type="checkbox"/>	J1-9
XCBUS028	<input type="checkbox"/>	J1-51	XCBUS078	<input type="checkbox"/>	J1-67
XCBUS029	<input type="checkbox"/>	J1-56	XCBUS079	<input type="checkbox"/>	J1-7
XCBUS030	<input type="checkbox"/>	J1-69	XCBUS080	<input type="checkbox"/>	J1-8
XCBUS031	<input type="checkbox"/>	J1-68	XCBUS083	<input type="checkbox"/>	J1-18
XCBUS032	<input type="checkbox"/>	J1-15	XCBUS084	<input type="checkbox"/>	J1-19
XCBUS034	<input type="checkbox"/>	J1-30	XCBUS085	<input type="checkbox"/>	J1-20
XCBUS037	<input type="checkbox"/>	J1-73	XCBUS086	<input type="checkbox"/>	J1-23
XCBUS038	<input type="checkbox"/>	J1-45	XCBUS087	<input type="checkbox"/>	J1-24
XCBUS039	<input type="checkbox"/>	J1-71	XCBUS088	<input type="checkbox"/>	J1-13
XCBUS040	<input type="checkbox"/>	J1-57	XCBUS093	<input type="checkbox"/>	J1-25
XCBUS041	<input type="checkbox"/>	J1-65	XCBUS094	<input type="checkbox"/>	J1-26
XCBUS042	<input type="checkbox"/>	J1-58	XCBUS106	<input type="checkbox"/>	J1-12
XCBUS043	<input type="checkbox"/>	J1-61	XCBUS109	<input type="checkbox"/>	J1-14
XCBUS044	<input type="checkbox"/>	J1-40	XCBUS111	<input type="checkbox"/>	J1-21
XCBUS046	<input type="checkbox"/>	J1-39	XCBUS142	<input type="checkbox"/>	J1-17
XCBUS047	<input type="checkbox"/>	J1-59		<input type="checkbox"/>	J1-11
XCBUS048	<input type="checkbox"/>	J1-60		<input type="checkbox"/>	J1-42
XCBUS049	<input type="checkbox"/>	J1-38		<input type="checkbox"/>	J1-43
XCBUS050	<input type="checkbox"/>	J1-78		<input type="checkbox"/>	J1-44
XCBUS051	<input type="checkbox"/>	J1-79		<input type="checkbox"/>	J1-46
XCBUS054	<input type="checkbox"/>	J1-82		<input type="checkbox"/>	J1-47
XCBUS056	<input type="checkbox"/>	J1-83		<input type="checkbox"/>	J1-48
XCBUS057	<input type="checkbox"/>	J1-35		<input type="checkbox"/>	J1-49
XCBUS058	<input type="checkbox"/>	J1-62		<input type="checkbox"/>	J1-63
XCBUS059	<input type="checkbox"/>	J1-66		<input type="checkbox"/>	J1-72
XCBUS060	<input type="checkbox"/>	J1-80		<input type="checkbox"/>	J1-74
XCBUS062	<input type="checkbox"/>	J1-81		<input type="checkbox"/>	J1-75
				<input type="checkbox"/>	J1-76

XCBUS[001:144]

COMPANY: XESS Corporation		
TITLE: XSA Board Prototyping Header		
DRAWN:	DATED:	REV: V1.2
RELEASED:	DATED:	SHEET: OF

Digilent FX2 Breadboard Reference Manual

Revision: September 26, 2006



www.digilentinc.com

215 E Main Suite D | Pullman, WA 99163
(509) 334 6306 Voice and Fax

Overview

The Digilent FX2 Breadboard (FX2BB) offers a ready-made solution for prototyping breadboarded or wire-wrapped circuits as accessories to Digilent system boards. The FX2BB provides connectors suitable for direct connection of various Digilent system boards and Digilent Pmod™ peripheral modules.

The FX2BB is available in a wire-wrap version or a solderless breadboard version.

Features include:

- two 630 tie point breadboards separated by 100 tie point bus strip (solderless breadboard version)
- 32x65 hole wire-wrap area (wire-wrap version)
- four 6-pin male header
- four 6-pin female header
- FX2 connector
- prototype/wire-wrap connections on every signal
- two power buses and one ground plane.

Functional Description

Power Connections

The FX2BB provides two power busses and a ground bus. The two power busses are labeled VU and VCC. These two busses are made available at each connector position on the board. There is also a ground plane that connects the ground pins from all connectors together.

The usual Digilent convention is to power the VCC bus at 3.3V and the VU bus at 5.0V. However depending on the system board connected and the power supply used, other

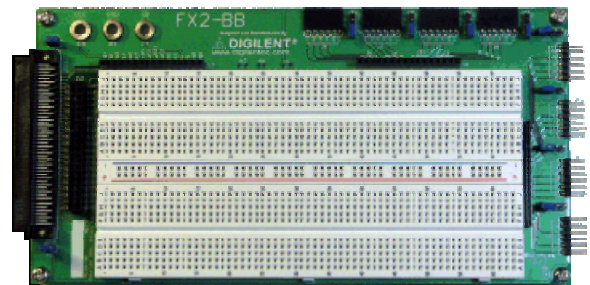


Figure 1
Digilent FX2 Breadboard

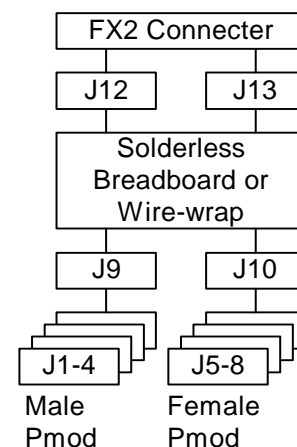


Figure 2
Block Diagram

voltages may be present. But observe caution before using any voltage other than 3.3V on the VCC bus. Most Digilent system boards will be damaged if the voltage on the VCC bus is greater than 3.3V.

Banana jacks J14-J16 provide connection points for connecting external, bench power supplies to the board to power the busses.

Alternatively, the power busses can be powered from the FX2 connector or any of the Pmod connectors. When configuring power jumpers and powering the board, it is important that each power supply bus be powered from a single power source. Damage can occur if the same bus is powered by more than one source.

Hirose, 100 Pin, FX2 Connector

FX2 connector J11 is provided on one side of the board for connection to Digilent system boards like the Nexys that contain an FX2 style connector. The Digilent FX2 connector signal convention provides for forty general-purpose I/O signals, three clock signals, JTAG signals, and power busses.

The forty general-purpose I/O signals from the FX2 connector are brought out to connector J12. These signals are labeled IO1-IO40. See Table 1 for a description of the relationship between FX2 connector pins and signal names on J12. The remaining signals from the FX2 connector are brought out to connector J13. See Table 1 for a description of the relationship between FX2 connector pins and connector J13 signal names.

In addition to the FX2 connector signals, connector J13 also provides access to the power and ground busses.

Jumper blocks JP9 and JP10 are used to connect or disconnect the VU and VCC busses of the system board and the VU and VCC busses on the FX2BB. Shorting blocks are placed on JP9 and/or JP10 to connect the busses, or removed to disconnect the busses.

Pmod Connectors

Digilent Pmod peripheral modules provide various peripheral functions. These can be as simple as buttons or switches for inputs and LEDs for outputs, to as complex as graphical

LCD display panels, accelerometers and keypads.

All Digilent Pmod modules use a six-wire interface for connection to a system board. The interface provides four I/O signals, power and ground. The signal definitions for the four signals as well as the voltage requirements for the power supply depend on the specific module.

The system board connection is through a 6-pin male connector. In addition to the system board connection, many Pmods, such as A/D and D/A converters, provide interfaces to outside signals. These connections are made through a 6-pin female connector.

The FX2BB provides two sets of four 6-pin connectors for connection of Pmods. Connectors J1-J4 are male connectors for connection to the external signal side of Pmods like A/D or D/A converters. Connectors J5-J8 are female connectors for connection to the system board side of Pmods.

The signals for Pmod connectors J1-J4 are brought out to connector J9. These signals are labeled; J1, 1-4; J2, 1-4, etc. Similarly, the signals for Pmod connectors J5-J8 are brought out to connector J10 and labeled; J5, 1-4, etc.

Each Pmod connector has an associated power select jumper. The power select jumper for J1 is JP1 and so on. These jumpers are used to select one of the two power busses on the FX2BB to provide power to the power supply pin on a Pmod plugged into that connector position. Placing a shorting block in the VCC position provides VCC power to the Pmod. Placing a shorting block in the VU position provides VU power to the Pmod. Place a shorting block so that it hangs off of the center pin only, disconnects power to the Pmod.

Table 1: FX2 Signals and Connector Pinout

A		B	
1	VCC	1	SHLD
2	VCC	2	GND
3	TMS	3	TDI (from host to peripheral)
4	JTSEL	4	TCK
5	TDO (From peripheral to host)	5	GND
6	IO1	6	GND
7	IO2	7	GND
8	IO3	8	GND
9	IO4	9	GND
10	IO5	10	GND
11	IO6	11	GND
12	IO7	12	GND
13	IO8	13	GND
14	IO9	14	GND
15	IO10	15	GND
16	IO11	16	GND
17	IO12	17	GND
18	IO13	18	GND
19	IO14	19	GND
20	IO15	20	GND
21	IO16	21	GND
22	IO17	22	GND
23	IO18	23	GND
24	IO19	24	GND
25	IO20	25	GND
26	IO21	26	GND
27	IO22	27	GND
28	IO23	28	GND
29	IO24	29	GND
30	IO25	30	GND
31	IO26	31	GND
32	IO27	32	GND
33	IO28	33	GND
34	IO29	34	GND
35	IO30	35	GND
36	IO31	36	GND
37	IO32	37	GND
38	IO33	38	GND
39	IO34	39	GND
40	IO35	40	GND
41	IO36	41	GND
42	IO37	42	GND
43	IO38	43	GND
44	IO39	44	GND



45	IO40	45	GND
46	GND	46	CLKIN (from peripheral to host)
47	CLKOUT (from host to peripheral)	47	GND
48	GND	48	CLKIO (from host to peripheral)
49	VU	49	VU
50	VU	50	SHLD

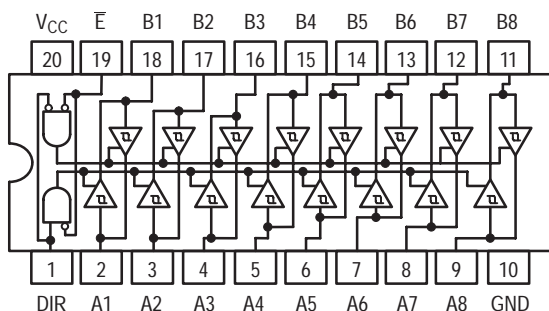
SN74LS245

Octal Bus Transceiver

The SN74LS245 is an Octal Bus Transmitter/Receiver designed for 8-line asynchronous 2-way data communication between data buses. Direction Input (DR) controls transmission of Data from bus A to bus B or bus B to bus A depending upon its logic level. The Enable input (\bar{E}) can be used to isolate the buses.

- Hysteresis Inputs to Improve Noise Immunity
- 2-Way Asynchronous Data Bus Communication
- Input Diodes Limit High-Speed Termination Effects
- ESD > 3500 Volts

LOGIC AND CONNECTION DIAGRAMS DIP (TOP VIEW)



TRUTH TABLE

INPUTS		OUTPUT
\bar{E}	DIR	
L	L	Bus B Data to Bus A
L	H	Bus A Data to Bus B
H	X	Isolation

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

GUARANTEED OPERATING RANGES

Symbol	Parameter	Min	Typ	Max	Unit
V_{CC}	Supply Voltage	4.75	5.0	5.25	V
T_A	Operating Ambient Temperature Range	0	25	70	°C
I_{OH}	Output Current – High			–3.0	mA
				–15	mA
I_{OL}	Output Current – Low			24	mA

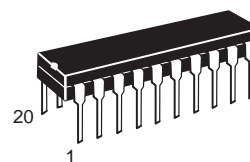


ON Semiconductor

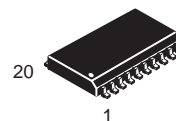
Formerly a Division of Motorola

<http://onsemi.com>

**LOW
POWER
SCHOTTKY**



**PLASTIC
N SUFFIX
CASE 738**



**SOIC
DW SUFFIX
CASE 751D**

ORDERING INFORMATION

Device	Package	Shipping
SN74LS245N	16 Pin DIP	1440 Units/Box
SN74LS245DW	16 Pin	2500/Tape & Reel

SN74LS245

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
V_{IH}	Input HIGH Voltage	2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage			0.8	V	Guaranteed Input LOW Voltage for All Inputs
$V_{T+}-V_{T-}$	Hysteresis	0.2	0.4		V	$V_{CC} = \text{MIN}$
V_{IK}	Input Clamp Diode Voltage		-0.65	-1.5	V	$V_{CC} = \text{MIN}$, $I_{IN} = -18 \text{ mA}$
V_{OH}	Output HIGH Voltage	2.4	3.4		V	$V_{CC} = \text{MIN}$, $I_{OH} = -3.0 \text{ mA}$
		2.0			V	$V_{CC} = \text{MIN}$, $I_{OH} = \text{MAX}$
V_{OL}	Output LOW Voltage		0.25	0.4	V	$I_{OL} = 12 \text{ mA}$, $V_{CC} = V_{CC} \text{ MIN}$, $V_{IN} = V_{IL} \text{ or } V_{IH}$ per Truth Table
			0.35	0.5	V	$I_{OL} = 24 \text{ mA}$
I_{OZH}	Output Off Current HIGH			20	μA	$V_{CC} = \text{MAX}$, $V_{OUT} = 2.7 \text{ V}$
I_{OZL}	Output Off Current LOW			-200	μA	$V_{CC} = \text{MAX}$, $V_{OUT} = 0.4 \text{ V}$
I_{IH}	Input HIGH Current	A or B, DR or \bar{E}		20	μA	$V_{CC} = \text{MAX}$, $V_{IN} = 2.7 \text{ V}$
		DR or \bar{E}		0.1	mA	$V_{CC} = \text{MAX}$, $V_{IN} = 7.0 \text{ V}$
		A or B		0.1	mA	$V_{CC} = \text{MAX}$, $V_{IN} = 5.5 \text{ V}$
I_{IL}	Input LOW Current			-0.2	mA	$V_{CC} = \text{MAX}$, $V_{IN} = 0.4 \text{ V}$
I_{OS}	Output Short Circuit Current (Note 1)	-40		-225	mA	$V_{CC} = \text{MAX}$
I_{CC}	Power Supply Current Total, Output HIGH			70	mA	$V_{CC} = \text{MAX}$
	Total, Output LOW			90		
	Total at HIGH Z			95		

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

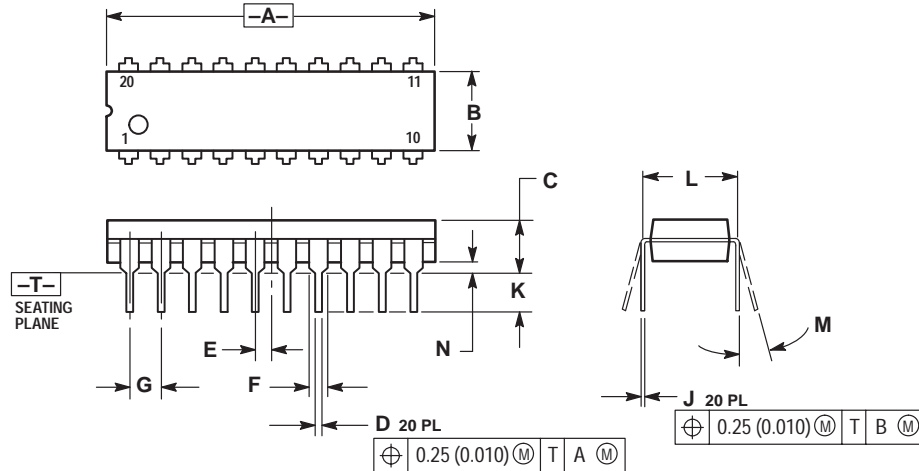
AC CHARACTERISTICS ($T_A = 25^\circ\text{C}$, $V_{CC} = 5.0 \text{ V}$, $T_{RISE}/T_{FALL} \leq 6.0 \text{ ns}$)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
t_{PLH} t_{PHL}	Propagation Delay, Data to Output		8.0 8.0	12 12	ns	$C_L = 45 \text{ pF}$, $R_L = 667 \Omega$
t_{PZH}	Output Enable Time to HIGH Level		25	40	ns	
t_{PZL}	Output Enable Time to LOW Level		27	40	ns	
t_{PLZ}	Output Disable Time from LOW Level		15	25	ns	$C_L = 5.0 \text{ pF}$, $R_L = 667 \Omega$
t_{PHZ}	Output Disable Time from HIGH Level		15	25	ns	

SN74LS245

PACKAGE DIMENSIONS

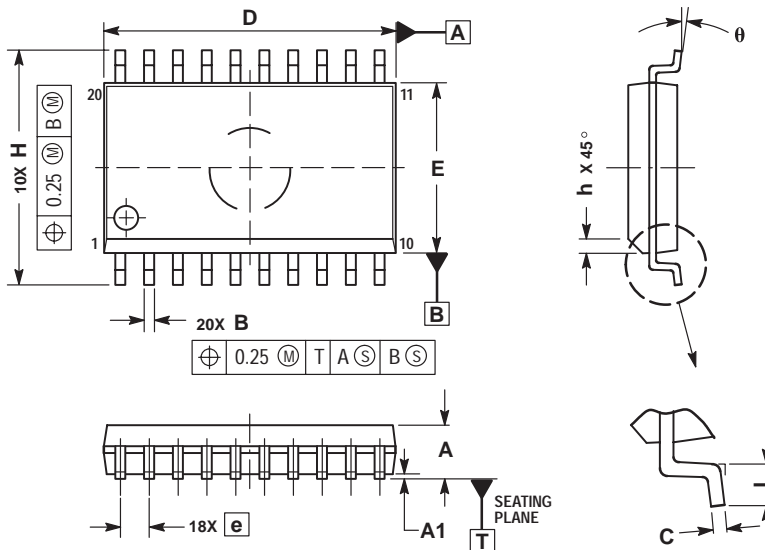
N SUFFIX PLASTIC PACKAGE CASE 738-03 ISSUE E



NOTES:

1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
4. DIMENSION B DOES NOT INCLUDE MOLD FLASH.


D SUFFIX PLASTIC SOIC PACKAGE CASE 751D-05 ISSUE F



NOTES:

1. DIMENSIONS ARE IN MILLIMETERS.
2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5M, 1994.
3. DIMENSIONS D AND E DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 PER SIDE.
5. DIMENSION B DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE PROTRUSION SHALL BE 0.13 TOTAL IN EXCESS OF B DIMENSION AT MAXIMUM MATERIAL CONDITION.

DIM	MILLIMETERS	
	MIN	MAX
A	2.35	2.65
A1	0.10	0.25
B	0.35	0.49
C	0.23	0.32
D	12.65	12.95
E	7.40	7.60
e	1.27 BSC	
H	10.05	10.55
h	0.25	0.75
L	0.50	0.90
θ	0°	7°

ON Semiconductor and  are trademarks of Semiconductor Components Industries, LLC (SCILLC). SCILLC reserves the right to make changes without further notice to any products herein. SCILLC makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does SCILLC assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. "Typical" parameters which may be provided in SCILLC data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. SCILLC does not convey any license under its patent rights nor the rights of others. SCILLC products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SCILLC product could create a situation where personal injury or death may occur. Should Buyer purchase or use SCILLC products for any such unintended or unauthorized application, Buyer shall indemnify and hold SCILLC and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that SCILLC was negligent regarding the design or manufacture of the part. SCILLC is an Equal Opportunity/Affirmative Action Employer.

PUBLICATION ORDERING INFORMATION

North America Literature Fulfillment:

Literature Distribution Center for ON Semiconductor
P.O. Box 5163, Denver, Colorado 80217 USA
Phone: 303-675-2175 or 800-344-3860 Toll Free USA/Canada
Fax: 303-675-2176 or 800-344-3867 Toll Free USA/Canada
Email: ONlit@hibbertco.com

N. American Technical Support: 800-282-9855 Toll Free USA/Canada

EUROPE: LDC for ON Semiconductor – European Support

German Phone: (+1) 303-308-7140 (M–F 2:30pm to 5:00pm Munich Time)
Email: ONlit-german@hibbertco.com
French Phone: (+1) 303-308-7141 (M–F 2:30pm to 5:00pm Toulouse Time)
Email: ONlit-french@hibbertco.com
English Phone: (+1) 303-308-7142 (M–F 1:30pm to 5:00pm UK Time)
Email: ONlit@hibbertco.com

ASIA/PACIFIC: LDC for ON Semiconductor – Asia Support

Phone: 303-675-2121 (Tue–Fri 9:00am to 1:00pm, Hong Kong Time)
Toll Free from Hong Kong 800-4422-3781
Email: ONlit-asia@hibbertco.com

JAPAN: ON Semiconductor, Japan Customer Focus Center

4-32-1 Nishi-Gotanda, Shinagawa-ku, Tokyo, Japan 141-8549
Phone: 81-3-5487-8345
Email: r14153@onsemi.com

Fax Response Line: 303-675-2167
800-344-3810 Toll Free USA/Canada

ON Semiconductor Website: <http://onsemi.com>

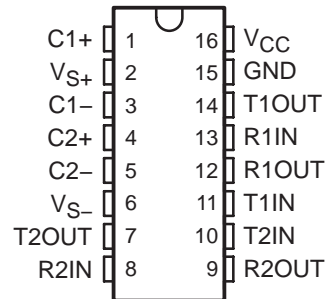
For additional information, please contact your local Sales Representative.

MAX232, MAX232I DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047L – FEBRUARY 1989 – REVISED MARCH 2004

- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Operates From a Single 5-V Power Supply With 1.0- μ F Charge-Pump Capacitors
- Operates Up To 120 kbit/s
- Two Drivers and Two Receivers
- ± 30 -V Input Levels
- Low Supply Current . . . 8 mA Typical
- ESD Protection Exceeds JESD 22 – 2000-V Human-Body Model (A114-A)
- Upgrade With Improved ESD (15-kV HBM) and 0.1- μ F Charge-Pump Capacitors is Available With the MAX202
- Applications
 - TIA/EIA-232-F, Battery-Powered Systems, Terminals, Modems, and Computers

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ± 30 -V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T _A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube of 25	MAX232N	MAX232N
	SOIC (D)	Tube of 40	MAX232D	MAX232
		Reel of 2500	MAX232DR	
	SOIC (DW)	Tube of 40	MAX232DW	MAX232
		Reel of 2000	MAX232DWR	
-40°C to 85°C	SOP (NS)	Reel of 2000	MAX232NSR	MAX232
	PDIP (N)	Tube of 25	MAX232IN	MAX232IN
	SOIC (D)	Tube of 40	MAX232ID	MAX232I
		Reel of 2500	MAX232IDR	
	SOIC (DW)	Tube of 40	MAX232IDW	MAX232I
		Reel of 2000	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

LinASIC is a trademark of Texas Instruments.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.



POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

Copyright © 2004, Texas Instruments Incorporated

MAX232, MAX232I
DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047L – FEBRUARY 1989 – REVISED MARCH 2004

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

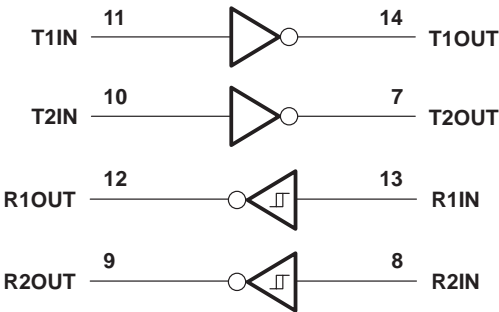
H = high level, L = low level

EACH RECEIVER

INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Input supply voltage range, V_{CC} (see Note 1)	–0.3 V to 6 V
Positive output supply voltage range, V_{S+}	$V_{CC} - 0.3$ V to 15 V
Negative output supply voltage range, V_{S-}	–0.3 V to –15 V
Input voltage range, V_I : Driver	–0.3 V to $V_{CC} + 0.3$ V
Receiver	±30 V
Output voltage range, V_O : T1OUT, T2OUT	$V_{S-} - 0.3$ V to $V_{S+} + 0.3$ V
R1OUT, R2OUT	–0.3 V to $V_{CC} + 0.3$ V
Short-circuit duration: T1OUT, T2OUT	Unlimited
Package thermal impedance, θ_{JA} (see Notes 2 and 3): D package	73°C/W
DW package	57°C/W
N package	67°C/W
NS package	64°C/W
Operating virtual junction temperature, T_J	150°C
Storage temperature range, T_{stg}	–65°C to 150°C

† Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

- NOTES: 1. All voltages are with respect to network GND.
2. Maximum power dissipation is a function of $T_{J(max)}$, θ_{JA} , and T_A . The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_{J(max)} - T_A)/\theta_{JA}$. Operating at the absolute maximum T_J of 150°C can affect reliability.
3. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions

		MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage	4.5	5	5.5	V
V_{IH}	High-level input voltage (T1IN, T2IN)	2			V
V_{IL}	Low-level input voltage (T1IN, T2IN)			0.8	V
R1IN, R2IN	Receiver input voltage			±30	V
T_A	Operating free-air temperature	MAX232	0	70	°C
		MAX232I	–40	85	

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (see Note 4 and Figure 4)

PARAMETER	TEST CONDITIONS	MIN	TYP‡	MAX	UNIT
I_{CC} Supply current	$V_{CC} = 5.5$ V, All outputs open, $T_A = 25^\circ\text{C}$		8	10	mA

‡ All typical values are at $V_{CC} = 5$ V and $T_A = 25^\circ\text{C}$.

NOTE 4: Test conditions are C1–C4 = 1 μF at $V_{CC} = 5$ V \pm 0.5 V.

MAX232, MAX232I

DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047L – FEBRUARY 1989 – REVISED MARCH 2004

DRIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (see Note 4)

PARAMETER		TEST CONDITIONS	MIN	TYP†	MAX	UNIT
V _{OH}	High-level output voltage	T1OUT, T2OUT R _L = 3 kΩ to GND	5	7		V
V _{OL}	Low-level output voltage‡	T1OUT, T2OUT R _L = 3 kΩ to GND		–7	–5	V
r _o	Output resistance	T1OUT, T2OUT V _{S+} = V _{S–} = 0, V _O = ±2 V	300			Ω
I _{OS} §	Short-circuit output current	T1OUT, T2OUT V _{CC} = 5.5 V, V _O = 0		±10		mA
I _{IS}	Short-circuit input current	T1IN, T2IN V _I = 0			200	μA

† All typical values are at V_{CC} = 5 V, T_A = 25°C.

‡ The algebraic convention, in which the least-positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.

§ Not more than one output should be shorted at a time.

NOTE 4: Test conditions are C1–C4 = 1 μF at V_{CC} = 5 V ± 0.5 V.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Note 4)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
SR	Driver slew rate	R _L = 3 kΩ to 7 kΩ, See Figure 2			30	V/μs
SR(t)	Driver transition region slew rate	See Figure 3		3		V/μs
	Data rate	One TOUT switching		120		kbit/s

NOTE 4: Test conditions are C1–C4 = 1 μF at V_{CC} = 5 V ± 0.5 V.

RECEIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (see Note 4)

PARAMETER		TEST CONDITIONS	MIN	TYP†	MAX	UNIT
V _{OH}	High-level output voltage	R1OUT, R2OUT I _{OH} = –1 mA	3.5			V
V _{OL}	Low-level output voltage‡	R1OUT, R2OUT I _{OL} = 3.2 mA			0.4	V
V _{IT+}	Receiver positive-going input threshold voltage	R1IN, R2IN V _{CC} = 5 V, T _A = 25°C		1.7	2.4	V
V _{IT–}	Receiver negative-going input threshold voltage	R1IN, R2IN V _{CC} = 5 V, T _A = 25°C	0.8	1.2		V
V _{hys}	Input hysteresis voltage	R1IN, R2IN V _{CC} = 5 V	0.2	0.5	1	V
r _i	Receiver input resistance	R1IN, R2IN V _{CC} = 5, T _A = 25°C	3	5	7	kΩ

† All typical values are at V_{CC} = 5 V, T_A = 25°C.

‡ The algebraic convention, in which the least-positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.

NOTE 4: Test conditions are C1–C4 = 1 μF at V_{CC} = 5 V ± 0.5 V.

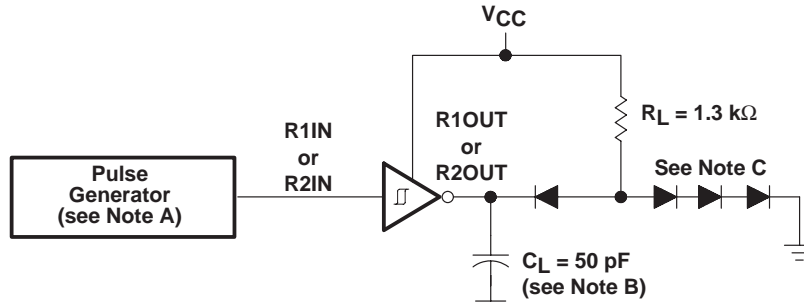
switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Note 4 and Figure 1)

PARAMETER		TYP	UNIT
t _{PLH(R)}	Receiver propagation delay time, low- to high-level output	500	ns
t _{PHL(R)}	Receiver propagation delay time, high- to low-level output	500	ns

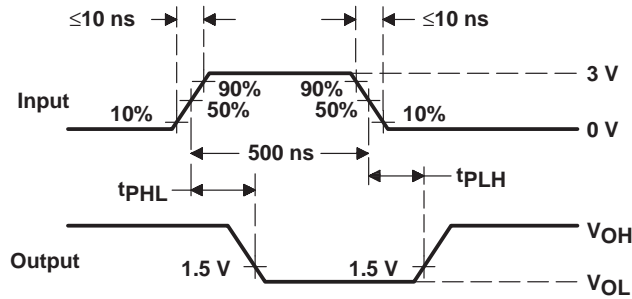
NOTE 4: Test conditions are C1–C4 = 1 μF at V_{CC} = 5 V ± 0.5 V.



PARAMETER MEASUREMENT INFORMATION



TEST CIRCUIT



WAVEFORMS

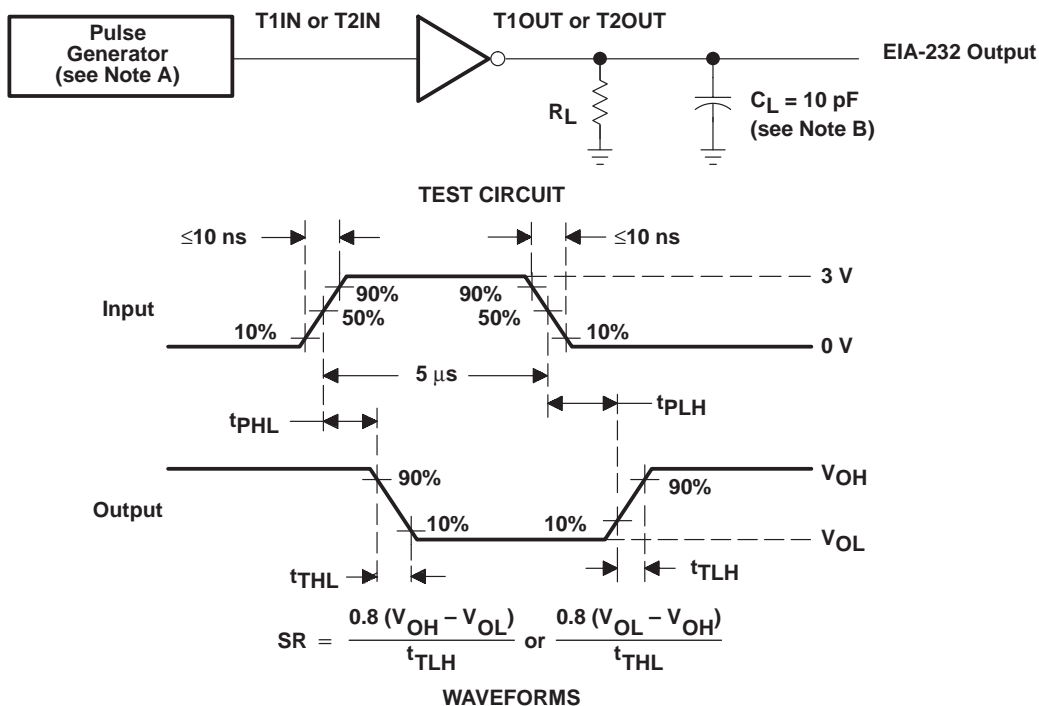
- NOTES: A. The pulse generator has the following characteristics: $Z_O = 50 \Omega$, duty cycle $\leq 50\%$.
B. C_L includes probe and jig capacitance.
C. All diodes are 1N3064 or equivalent.

Figure 1. Receiver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements

MAX232, MAX232I DUAL EIA-232 DRIVERS/RECEIVERS

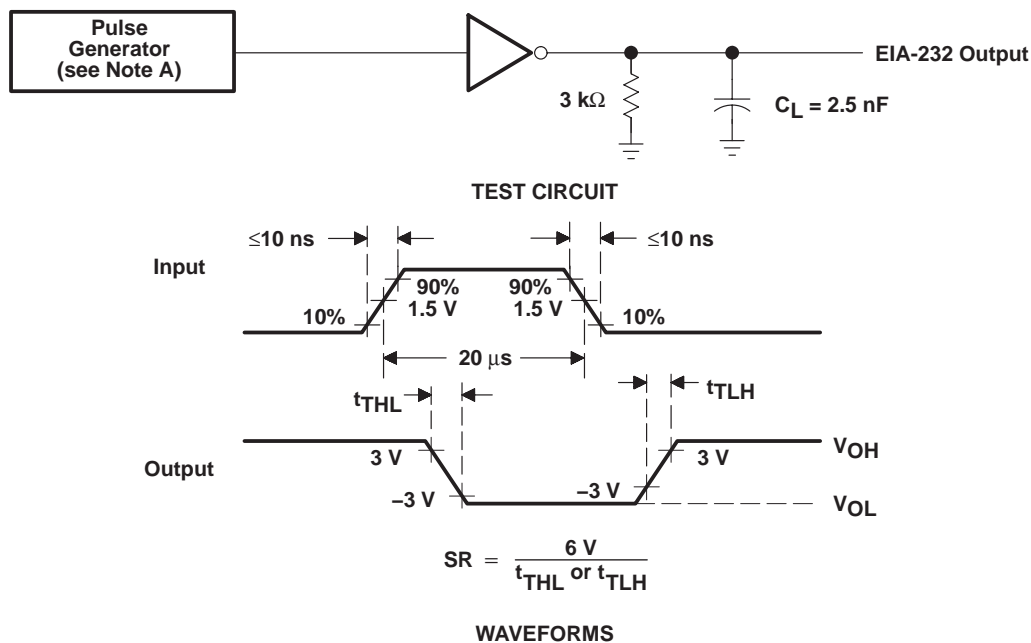
SLLS047L – FEBRUARY 1989 – REVISED MARCH 2004

PARAMETER MEASUREMENT INFORMATION



NOTES: A. The pulse generator has the following characteristics: $Z_O = 50 \Omega$, duty cycle $\leq 50\%$.
B. C_L includes probe and jig capacitance.

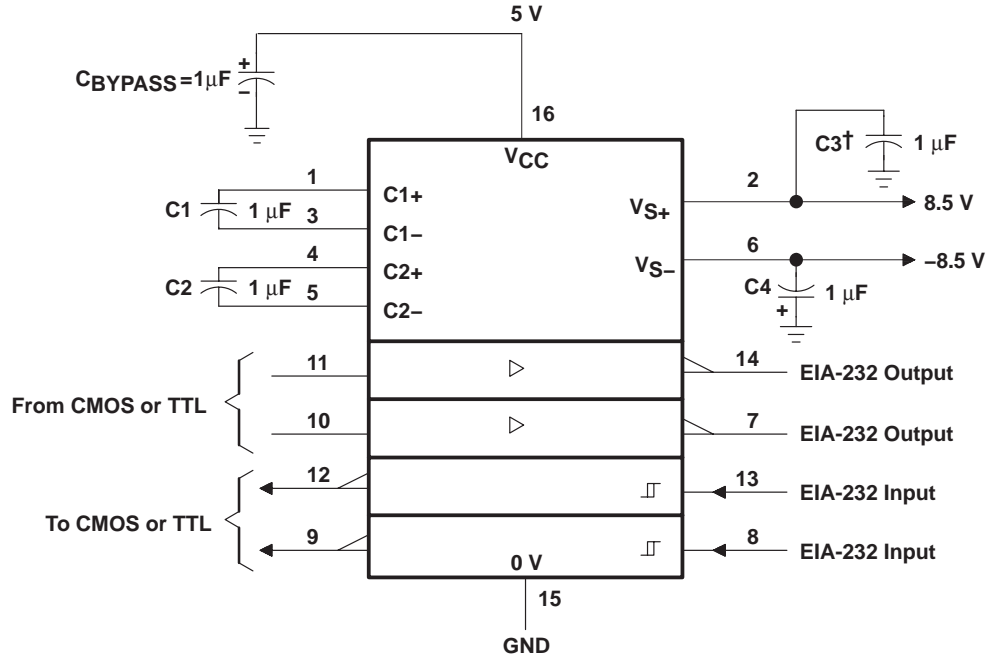
Figure 2. Driver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements (5- μs Input)



NOTE A: The pulse generator has the following characteristics: $Z_O = 50 \Omega$, duty cycle $\leq 50\%$.

Figure 3. Test Circuit and Waveforms for t_{THL} and t_{TLH} Measurements (20- μs Input)

APPLICATION INFORMATION



† C3 can be connected to V_{CC} or GND.

NOTES: A. Resistor values shown are nominal.

B. Nonpolarized ceramic capacitors are acceptable. If polarized tantalum or electrolytic capacitors are used, they should be connected as shown. In addition to the 1- μ F capacitors shown, the MAX202 can operate with 0.1- μ F capacitors.

Figure 4. Typical Operating Circuit

PACKAGING INFORMATION

Orderable Device	Status ⁽¹⁾	Package Type	Package Drawing	Pins	Package Qty	Eco Plan ⁽²⁾	Lead/Ball Finish	MSL Peak Temp ⁽³⁾
MAX232D	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DE4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DG4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DR	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DRE4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DRG4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DW	ACTIVE	SOIC	DW	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DWE4	ACTIVE	SOIC	DW	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DWG4	ACTIVE	SOIC	DW	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DWR	ACTIVE	SOIC	DW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DWRE4	ACTIVE	SOIC	DW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232DWRG4	ACTIVE	SOIC	DW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232ID	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDE4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDG4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDR	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDRE4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDRG4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDW	ACTIVE	SOIC	DW	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDWE4	ACTIVE	SOIC	DW	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDWG4	ACTIVE	SOIC	DW	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDWR	ACTIVE	SOIC	DW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDWRE4	ACTIVE	SOIC	DW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IDWRG4	ACTIVE	SOIC	DW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232IN	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type

Orderable Device	Status ⁽¹⁾	Package Type	Package Drawing	Pins	Package Qty	Eco Plan ⁽²⁾	Lead/Ball Finish	MSL Peak Temp ⁽³⁾
MAX232INE4	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type
MAX232N	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type
MAX232NE4	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type
MAX232NSR	ACTIVE	SO	NS	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232NSRE4	ACTIVE	SO	NS	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
MAX232NSRG4	ACTIVE	SO	NS	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM

⁽¹⁾ The marketing status values are defined as follows:

ACTIVE: Product device recommended for new designs.

LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.

NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.

PREVIEW: Device has been announced but is not in production. Samples may or may not be available.

OBSOLETE: TI has discontinued the production of the device.

⁽²⁾ Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check <http://www.ti.com/productcontent> for the latest availability information and additional product content details.

TBD: The Pb-Free/Green conversion plan has not been defined.

Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.

Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.

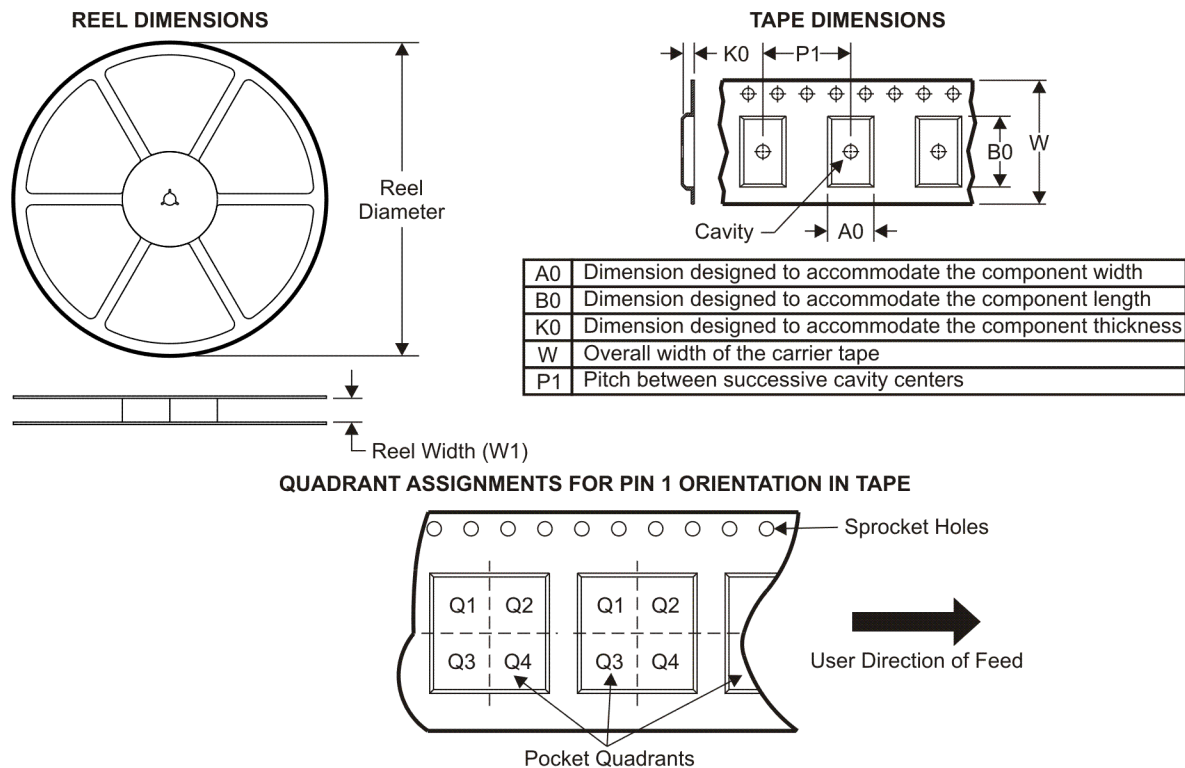
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

⁽³⁾ MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

Important Information and Disclaimer: The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

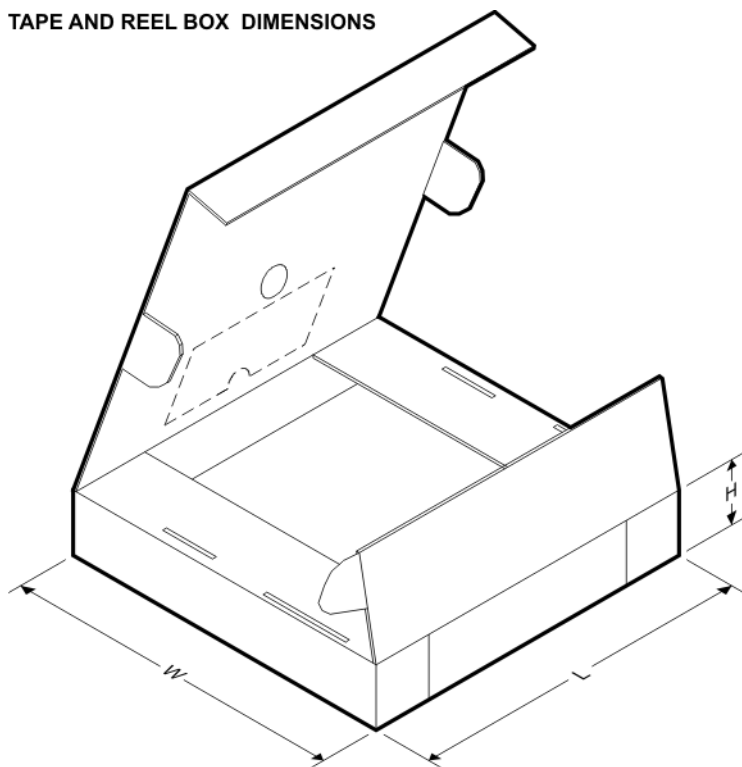
TAPE AND REEL INFORMATION



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
MAX232DR	SOIC	D	16	2500	330.0	16.4	6.5	10.3	2.1	8.0	16.0	Q1
MAX232DR	SOIC	D	16	2500	330.0	16.4	6.5	10.3	2.1	8.0	16.0	Q1
MAX232DWR	SOIC	DW	16	2000	330.0	16.4	10.75	10.7	2.7	12.0	16.0	Q1
MAX232IDR	SOIC	D	16	2500	330.0	16.4	6.5	10.3	2.1	8.0	16.0	Q1
MAX232IDWR	SOIC	DW	16	2000	330.0	16.4	10.75	10.7	2.7	12.0	16.0	Q1
MAX232NSR	SO	NS	16	2000	330.0	16.4	8.2	10.5	2.5	12.0	16.0	Q1

TAPE AND REEL BOX DIMENSIONS



*All dimensions are nominal

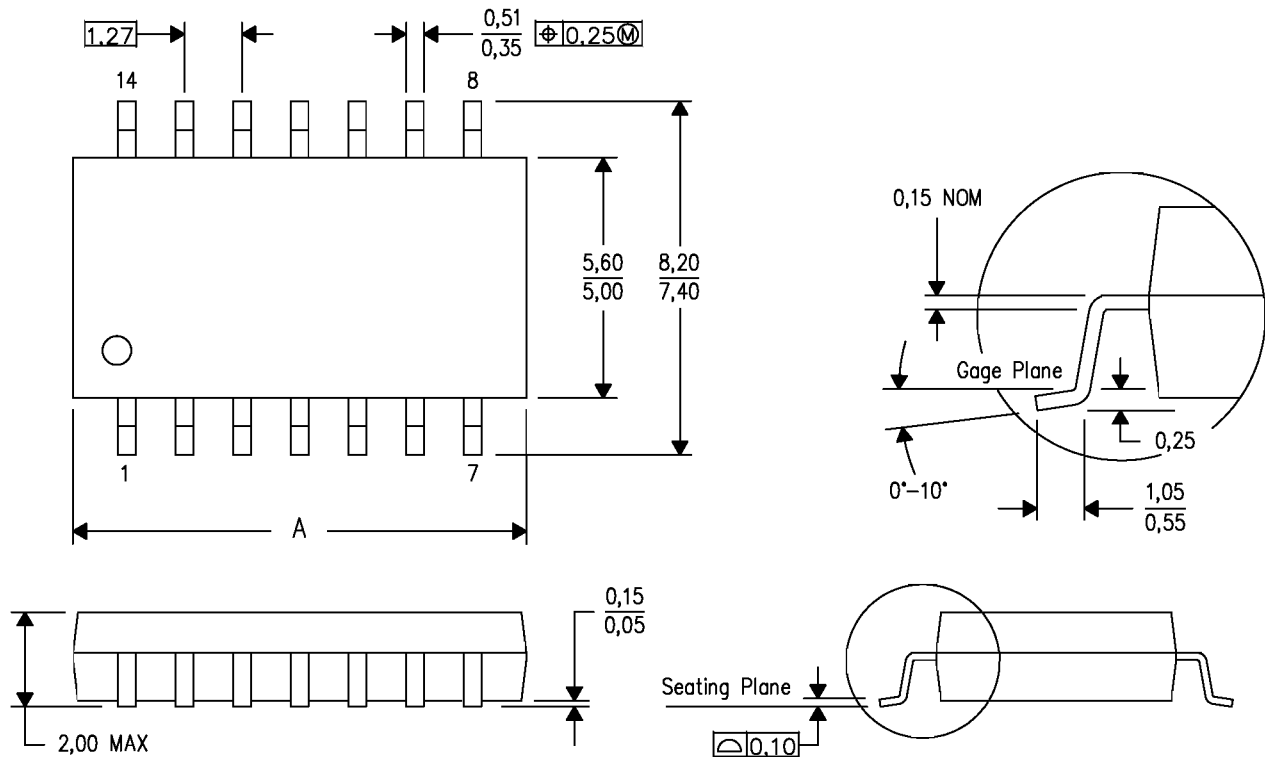
Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
MAX232DR	SOIC	D	16	2500	346.0	346.0	33.0
MAX232DR	SOIC	D	16	2500	333.2	345.9	28.6
MAX232DWR	SOIC	DW	16	2000	346.0	346.0	33.0
MAX232IDR	SOIC	D	16	2500	333.2	345.9	28.6
MAX232IDWR	SOIC	DW	16	2000	346.0	346.0	33.0
MAX232NSR	SO	NS	16	2000	346.0	346.0	33.0

MECHANICAL DATA

NS (R-PDSO-G**)

PLASTIC SMALL-OUTLINE PACKAGE

14-PINS SHOWN



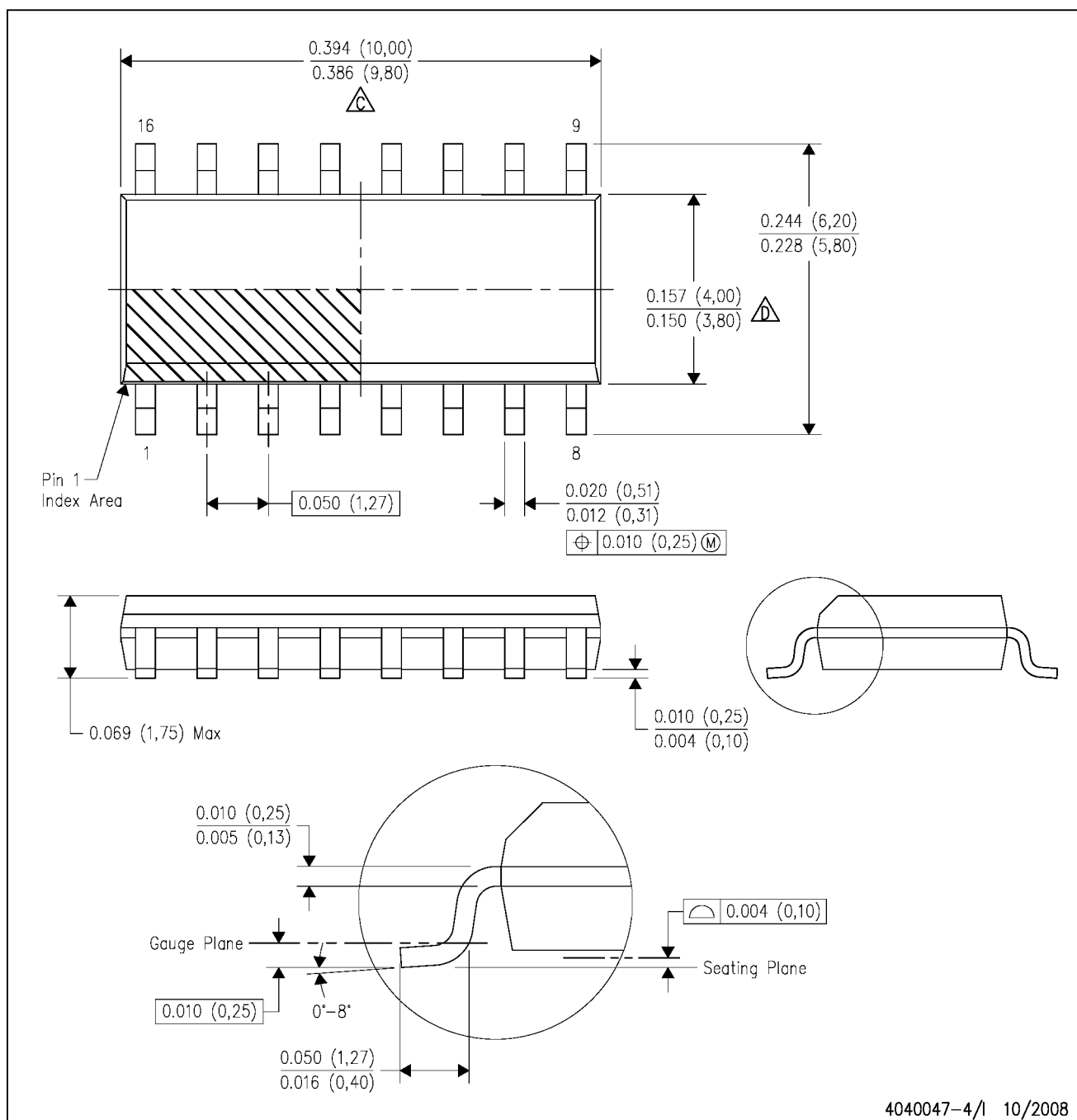
DIM \ PINS **	14	16	20	24
A MAX	10,50	10,50	12,90	15,30
A MIN	9,90	9,90	12,30	14,70

4040062/C 03/03

- NOTES:
- A. All linear dimensions are in millimeters.
 - B. This drawing is subject to change without notice.
 - C. Body dimensions do not include mold flash or protrusion, not to exceed 0,15.

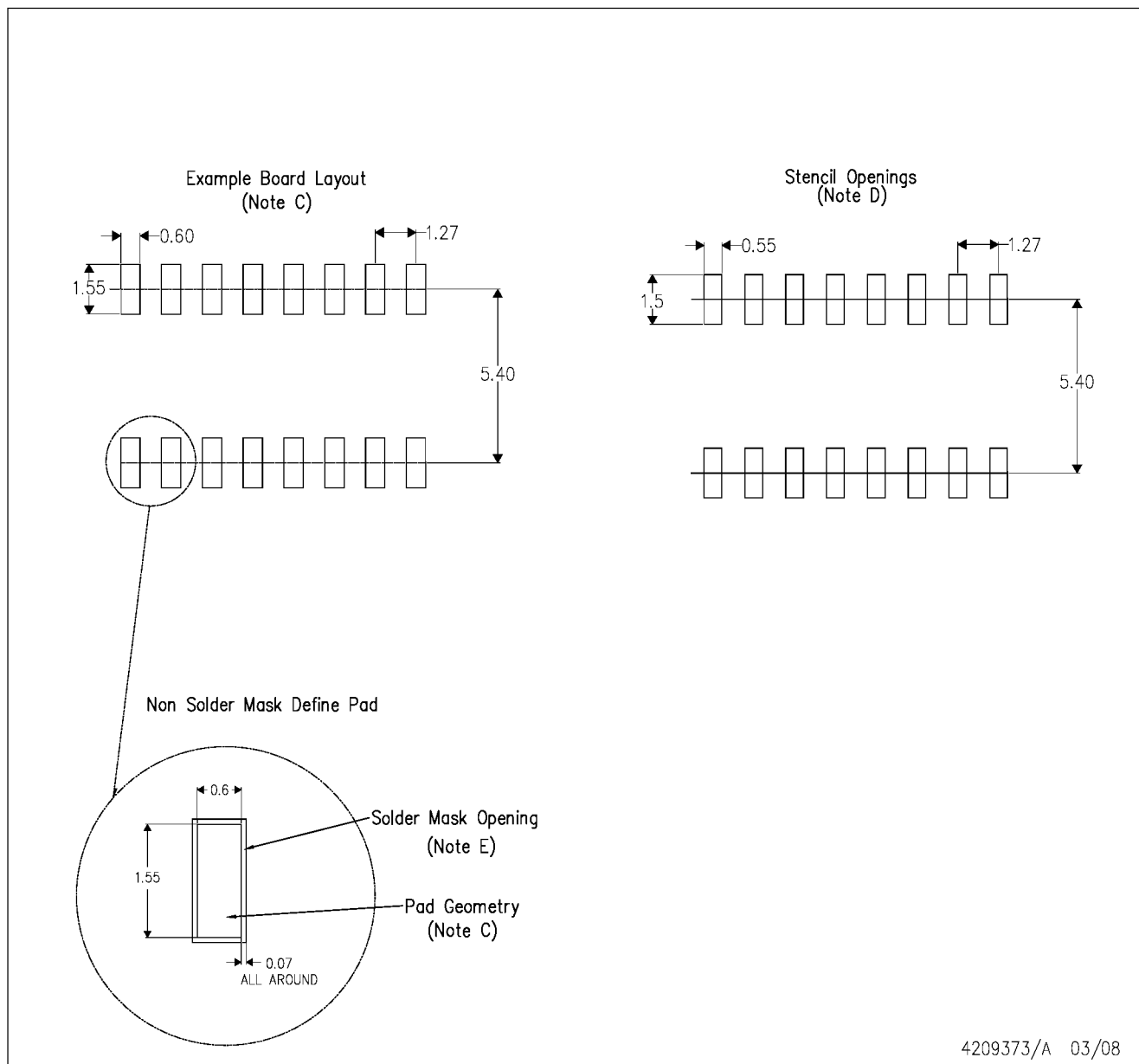
D (R-PDSO-G16)

PLASTIC SMALL-OUTLINE PACKAGE



- NOTES:
- A. All linear dimensions are in inches (millimeters).
 - B. This drawing is subject to change without notice.
 - \triangle Body length does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed .006 (0,15) per end.
 - \triangle Body width does not include interlead flash. Interlead flash shall not exceed .017 (0,43) per side.
 - E. Reference JEDEC MS-012 variation AC.

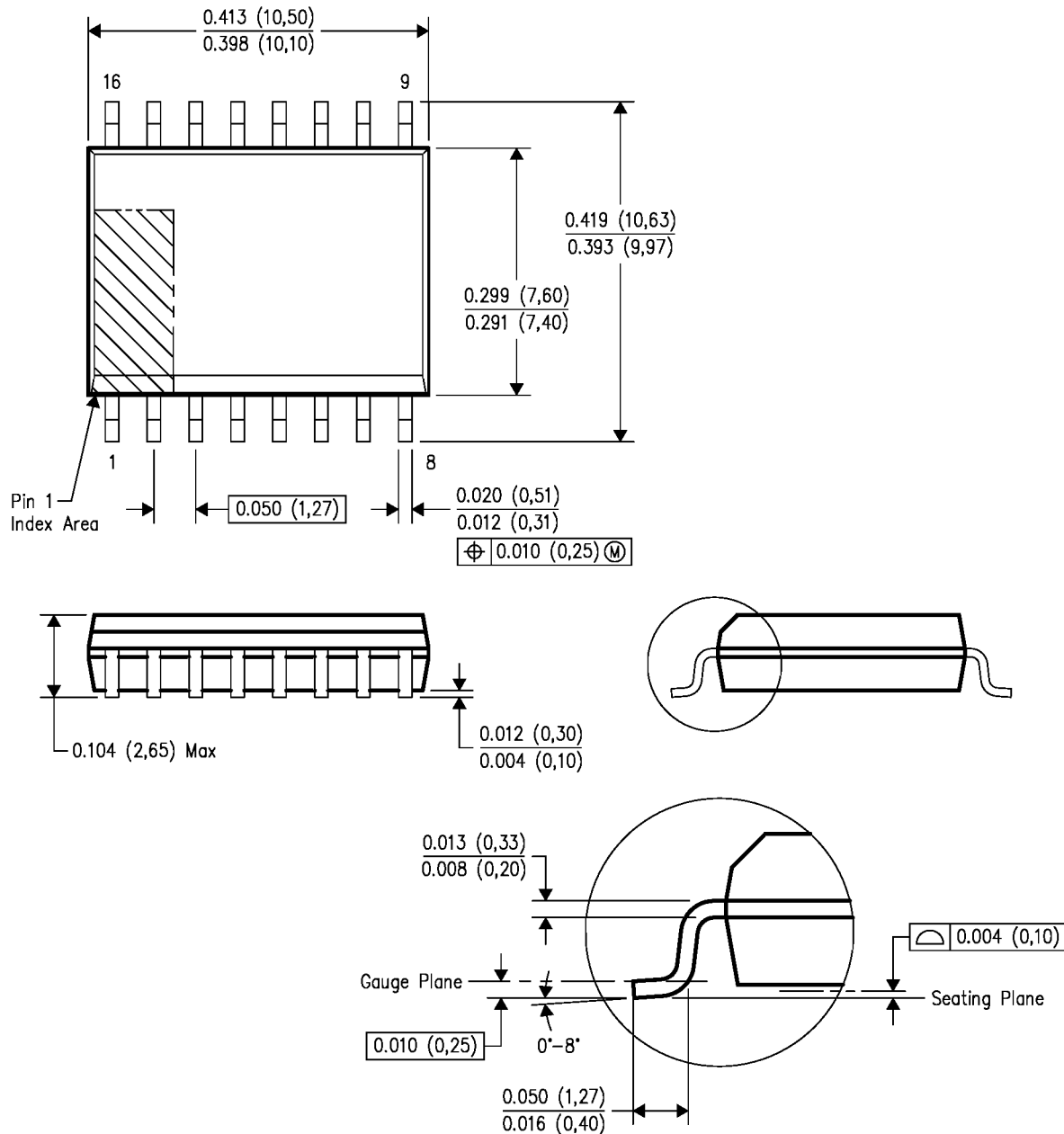
D(R-PDSO-G16)



- NOTES:
- A. All linear dimensions are in millimeters.
 - B. This drawing is subject to change without notice.
 - C. Refer to IPC7351 for alternate board design.
 - D. Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Refer to IPC-7525
 - E. Customers should contact their board fabrication site for solder mask tolerances between and around signal pads.

DW (R-PDSO-G16)

PLASTIC SMALL-OUTLINE PACKAGE



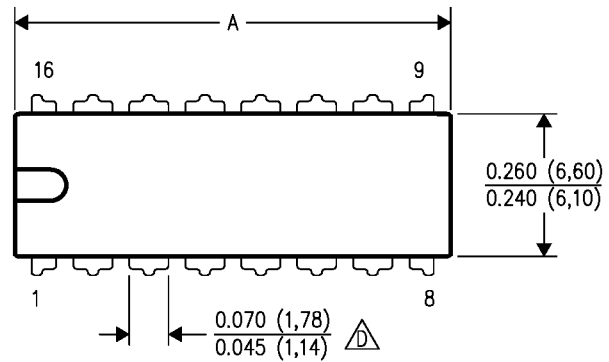
4040000-2/F 06/2004

- NOTES:
- A. All linear dimensions are in inches (millimeters).
 - B. This drawing is subject to change without notice.
 - C. Body dimensions do not include mold flash or protrusion not to exceed 0.006 (0,15).
 - D. Falls within JEDEC MS-013 variation AA.

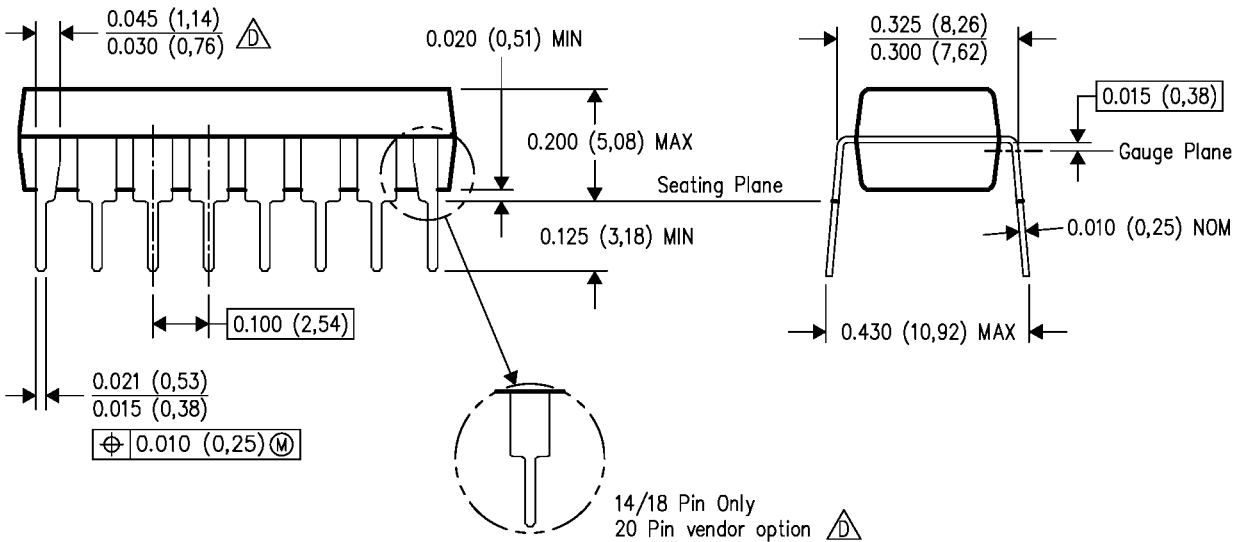
N (R-PDIP-T**)

16 PINS SHOWN

PLASTIC DUAL-IN-LINE PACKAGE



PINS **	14	16	18	20
DIM				
A MAX	0.775 (19,69)	0.775 (19,69)	0.920 (23,37)	1.060 (26,92)
A MIN	0.745 (18,92)	0.745 (18,92)	0.850 (21,59)	0.940 (23,88)
MS-001 VARIATION	AA	BB	AC	AD



4040049/E 12/2002

- NOTES:
- A. All linear dimensions are in inches (millimeters).
 - B. This drawing is subject to change without notice.
 - C. Falls within JEDEC MS-001, except 18 and 20 pin minimum body length (Dim A).
 - D. The 20 pin end lead shoulder width is a vendor option, either half or full width.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products

Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
RF/IF and ZigBee® Solutions	www.ti.com/lprf

Applications

Audio	www.ti.com/audio
Automotive	www.ti.com/automotive
Broadband	www.ti.com/broadband
Digital Control	www.ti.com/digitalcontrol
Medical	www.ti.com/medical
Military	www.ti.com/military
Optical Networking	www.ti.com/opticalnetwork
Security	www.ti.com/security
Telephony	www.ti.com/telephony
Video & Imaging	www.ti.com/video
Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2008, Texas Instruments Incorporated

Bibliography:

1. “Wireless Communications Principles & Practice” by Theodore S. Rappaport, Prentice-Hall PTR.
2. Microprocessor and PC Interfacing, Programming and Hardware, Hall D.V.
3. The 8051 microcontroller and Embedded System, Mohommad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. McKinley, Second Edition.
4. VHDL Programming by Example, Douglas L. Perry -4th Ed McGraw Hill
5. Circuit Design with VHDL, Volnei A. Pedroni, First Edition.
6. Course A CDMA Basic Theory (c11), ZTE Corporation
7. Past year projects in CDMA (Data Transfer via CDMA- 059/BEX batch)
8. <http://www.wikipedia.org>

Credits:

- ✚ Prof. Dr. Shashidhar Ram Joshi, Head of Department ,DOECE, Pulchowk Campus
- ✚ Dr. Surendra Shrestha, Project Co-coordinator, DOECE, Pulchowk Campus
- ✚ Mr. Lochan Lal Amatya, Manager, IT Directorate, Nepal Telecom (External Examiner)
- ✚ Mr. Sudip Rimal , Internal Examiner, DOECE, Pulchowk Campus
- ✚ Prof. Dr. Dinesh Kumar Sharma ,Project Supervisor, DOECE, Pulchowk Campus
- ✚ Mr. Pradeep Poudyal, Project Supervisor, DOECE, Pulchowk Campus

Contact List:

- ✚ **Rikesh Shakya (061/bex/434)**
Gabahal, Patan
9841676084
rikesh_eikir@hotmail.com
- ✚ **Sabin Maharjan (061/bex/436)**
Siddhipur, VDC, Ward No. 9, Lalitpur
9841444833
maharjan_sabins@hotmail.com
- ✚ **Sudat Tuladhar (061/bex/441)**
Ombahal, NewRoad, Kathmandu
9841783167
sudat_tul@hotmail.com
- ✚ **Sujan Raj Shrestha (061/bex/443)**
Bhimsensthan, Kathmandu
9841774912
rshr01sujan@hotmail.com

Source Codes and Soft Copy available at:

<http://groups.google.com/group/pulchowk-061bex>



Project Members of “CDMA Based Personal Communication System”

Sujan Raj Shrestha	(061/bex/443)
Rikesh Shakya	(061/bex/434)
Sabin Maharjan	(061/bex/436)
Sudat Tuladhar	(061/bex/441)